Correspondence

New Techniques for Constructing EC/AUED Codes

Jehoshua Bruck and Mario Blaum

***Abstract*—The most common method to construct a t-error correcting/all unidirectional error detecting (EC/AUED) code is to choose a t-error correcting (EC) code and then to append a tail in such a way that the new code can detect more than t errors when they are unidirectional. The tail is a function of the weight of the codeword.**

We present two new techniques for constructing t-EC/AUED codes. The first technique modifies the t-EC code in such a way that the weight distribution of the original code is reduced. So, a smaller tail is needed. Frequently, this technique gives less overall redundancy than the best available t-EC/AUED codes.

The second technique improves the parameters of the tails with respect to previous results.

***Index Terms*—Coset, decoding, descending tail matrix, encoding, error-correcting codes, redundancy, unidirectional errors, unidirectional error detecting codes.**

I. Introduction

The problem of finding error correcting/all unidirectional error detecting codes (EC/AUED) has received wide attention in recent literature [1]–[10]. The reader can find good discussions about the practical applications of EC/AUED codes in most of the references cited above.

In this paper, we concentrate on the mathematical aspects of the codes. Given two binary vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of length n, denote by $N(\boldsymbol{u},\boldsymbol{v})$ the number of $1 \rightarrow 0$ transitions from $\boldsymbol{u}$ to $\boldsymbol{v}$. (For example, if $\boldsymbol{u} = 10101$ and $\boldsymbol{v} = 00011$, then $N(\boldsymbol{u},\boldsymbol{v}) = 2$ and $N(\boldsymbol{v},\boldsymbol{u}) = 1$). Clearly, $d_H(\boldsymbol{u},\boldsymbol{v}) = N(\boldsymbol{u},\boldsymbol{v}) + N(\boldsymbol{v},\boldsymbol{u})$, where d_H denotes Hamming distance.

We say that $\boldsymbol{u}$ is contained in $\boldsymbol{v}$ ($\boldsymbol{u} \subseteq \boldsymbol{v}$) if $N(\boldsymbol{u},\boldsymbol{v}) = 0$.

Assume that $\boldsymbol{u}$ is transmitted but $\hat{\boldsymbol{u}}$ is received. We say that $\boldsymbol{u}$ has suffered unidirectional errors if either $\boldsymbol{u} \subseteq \hat{\boldsymbol{u}}$ or $\hat{\boldsymbol{u}} \subseteq \boldsymbol{u}$.

We are interested in codes that can correct up to t errors and detect all unidirectional errors when the number of unidirectional errors is greater than t. In other words, when more than t errors occur and those errors are unidirectional, we do not want the received word to fall into a sphere of radius t whose center is a codeword. The next theorem [4], [5] gives necessary and sufficient conditions for a code to be t-EC/AUED.

Theorem 1.1: Let C be a subset of $\{0,1\}^n$. Then C is a t-EC/AUED code if and only if, for any pair of vectors $\boldsymbol{u} \in C$ and $\boldsymbol{v} \in C$, $N(\boldsymbol{u},\boldsymbol{v}) \geq t+1$ and $N(\boldsymbol{v},\boldsymbol{u}) \geq t+1$.

Given k information bits, the way most authors construct a t-EC/AUED code C of length n is as follows: first the information bits are encoded into a t-EC (error-correcting) code C' of length n', with n' as small as possible. Usually but not necessarily, this code is systematic. Choosing a good $[n', k, 2t+1]$ code is not a problem. There is a vast literature on the subject. For instance, we can use the tables in [11] to choose, given k and t, the best k-dimensional t-EC code known.

Manuscript received May 15, 1989; revised July 10, 1990.

The authors are with IBM Research Division, Almaden Research Center, San Jose, CA 95120.

IEEE Log Number 9103101.

The second step involves adding a tail of length r as further redundancy. The length of the code is then $n = n' + r$, and the total redundancy is $n' - k + r$. The tail is a function of the weight of the vector. The goal is to obtain a tail with r as small as possible.

For the sake of completeness, we give a general construction for t-EC/AUED codes. We need a definition first. We denote by $\lceil x \rceil$ ($\lfloor x \rfloor$) the smallest (largest) integer j such that $j \geq x$ ($j \leq x$).

Definition 1.1: A descending tail matrix of strength s is an $m \times r$ $\{0,1\}$-matrix with rows $\boldsymbol{t}_i$, $0 \leq i \leq m-1$, such that for all $0 \leq i \leq j \leq m-1$,

$$N(\boldsymbol{t}_i, \boldsymbol{t}_j) \geq \min\{s, \lceil (j-i)/2 \rceil\}.$$

An $m \times r$ matrix of strength s is denoted $T(m, r; s)$.

Construction 1.1: Let C' be a t-EC of length n' and let T be a $T(n'+1, r; t+1)$ descending tail matrix with rows $\boldsymbol{t}_0, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_{n'}$. Let C be the following code of length $n'+r$:

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

where $w(\boldsymbol{v})$ denotes the Hamming weight of $\boldsymbol{v}$. Then C is a t-EC/AUED code.

Proving that C is a t-EC/AUED is relatively easy using Theorem 1.1 [1].

Some of the best descending tail matrices are given in [1]. As said before, the goal is to make r as small as possible. The construction in [1] heavily depends on the best asymmetric error-correcting codes available.

In this paper, we propose two different techniques to reduce the redundancy of t-EC/AUED codes. The two methods can be used together. The first one involves using t-EC codes that contain the all-1 vector (for instance, BCH codes and the Golay code have this property). When choosing a codeword, we take either a codeword or its complement, according to which of the two has smaller weight. We have to pay a bit for this operation, but the weight distribution is reduced by half. We then append a tail in the way described by Construction 1.1. Overall, we will often gain in redundancy.

The construction will be described in detail in Section II. We then discuss the problems of encoding and decoding in Section III. Although the new codes are not strictly systematic, they are very close to being so. We will see that encoding and decoding are nearly as simple as in the systematic case. In Section IV, we provide tables and examples.

Section V can be read independently of the rest of the paper. There, we provide some techniques to improve upon the tail matrices given in [1].

II. Construction

As stated in the Introduction, the construction in [1] depends on a tail that is appended to each codeword in a t-EC code. This tail is a function of the weight of the codeword and it is obtained from a descending tail matrix of strength s (Definition 1.1). Table I gives a list of parameters for the descending tail matrices obtained in [1].

The next construction is the main result in this seciton. It can be viewed as a modification of Construction 1.1.

TABLE I
PARAMETERS OF SOME DESCENDING TAIL MATRICES $T(m, r; t+1)$

t	r	m	t	r	m	t	r	m	t	r	m
1	2	4	2	3	6	3	4	8	4	5	10
1	3	6	2	4	8	3	5	10	4	6	12
1	4	8	2	5	10	3	6	12	4	7	14
1	5	12	2	6	12	3	7	14	4	8	16
1	6	16	2	7	16	3	8	16	4	9	18
1	7	24	2	8	20	3	9	20	4	10	20
1	8	48	2	9	24	3	10	24	4	11	24
1	9	72	2	10	32	3	11	28	4	12	28
1	10	144	2	11	48	3	12	32	4	13	32
1	11	248	2	12	72	3	13	40	4	14	36
1	12	432	2	13	120	3	14	48	4	15	40
			2	14	216	3	15	72	4	16	48
			2	15	392	3	16	120	4	17	56
						3	17	180	4	18	72
						3	18	264	4	19	104
						3	19	488	4	20	156
									4	21	216
									4	22	288

Construction 2.1: Let k be the number of information bits. Assume that we want to construct a t-EC/AUED code. Then:

1) Choose an $[n', k+1, d]$ EC code $(d \geq 2t+1)$ C' containing the all-1 vector with n' as small as possible.
2) Choose a $T(\lfloor n'/2 \rfloor + 1, r; t+1)$ descending tail matrix T with rows $\boldsymbol{t}_i$, $0 \leq i \leq \lfloor n'/2 \rfloor$ and r as small as possible.
3) Let C be the code

$$C = \{(\boldsymbol{c}, \boldsymbol{t}_{w(\boldsymbol{c})}) : \boldsymbol{c} \in C', w(\boldsymbol{c}) \leq n'/2\}.$$

The code C obtained in the previous construction is t-EC/AUED since the subset of codewords of weight $\leq \lfloor n/2 \rfloor$ is still a t-EC code. According to Construction 1.1, the tail makes it t-EC/AUED.

Example 2.1: Assume $k = 3$ and $t = 1$. According to Construction 2.1, we consider the $[7, 4, 3]$ Hamming code whose generator matrix is

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

We easily see that the codewords of weight ≤ 3 are

$$\begin{aligned} \boldsymbol{c}_0 &= 0000000 \\ \boldsymbol{c}_1 &= 1000011 \\ \boldsymbol{c}_2 &= 0100101 \\ \boldsymbol{c}_3 &= 0010110 \\ \boldsymbol{c}_4 &= 1001100 \\ \boldsymbol{c}_5 &= 0101010 \\ \boldsymbol{c}_6 &= 0011001 \\ \boldsymbol{c}_7 &= 1110000. \end{aligned}$$

According to [1], we can use the $T(4, 2; 2)$ matrix

$$T = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

The code is then given by the following set of codewords:

$$\begin{aligned} \boldsymbol{v}_0 &= 0000000 \quad 11 \\ \boldsymbol{v}_1 &= 1000011 \quad 00 \\ \boldsymbol{v}_2 &= 0100101 \quad 00 \\ \boldsymbol{v}_3 &= 0010110 \quad 00 \\ \boldsymbol{v}_4 &= 1001100 \quad 00 \\ \boldsymbol{v}_5 &= 0101010 \quad 00 \\ \boldsymbol{v}_6 &= 0011001 \quad 00 \\ \boldsymbol{v}_7 &= 1110000 \quad 00. \end{aligned}$$

Notice that we have 3 information bits and 6 redundant bits. If we use the construction in [1], we need 7 redundant bits.

Sometimes, taking a coset instead of the code itself may be convenient to reduce the span of the weight distribution.

III. ENCODING AND DECODING

In the previous section we described a t-EC/AUED code but we did not explain how to encode the data. This is very easily done, as we will see.

Assume we want to encode k bits into a t-EC/AUED code C. Choose an $[n', k+1, 2t+1]$ code C' containing the all-1 vector (with n' as small as possible) and a $T(\lfloor n'/2 \rfloor + 1, r, t+1)$ descending tail matrix T (with r as small as possible). The symbol $\oplus$ denotes "exclusive-OR" and 1 denotes the all-1 vector. Then proceed as follows:

Algorithm 3.1 (Encoding Algorithm) Let $\boldsymbol{u} = (u_1, u_2, \cdots, u_k)$ be the vector of information bits. Then:

1) Encode $(\boldsymbol{u}, 0) = (u_1, u_2, \cdots, u_k, 0)$ into a vector $\boldsymbol{c}$ in C'.
2) If $w(\boldsymbol{c}) > \lfloor n'/2 \rfloor$ then $\boldsymbol{c} \leftarrow \boldsymbol{c} \oplus 1$.
3) Let $\boldsymbol{v} = (\boldsymbol{c}, \boldsymbol{t}_{w(\boldsymbol{c})})$ be the output of the encoder, where $\boldsymbol{t}_i$, $0 \leq i \leq \lfloor n'/2 \rfloor$, are the rows of T.

Observe that code C' is not required to be systematic. However, if that is the case, the t-EC/AUED code C will be practically systematic, in the sense that the first k bits in codeword $\boldsymbol{v}$ will either be the information bits or their complements.

Example 3.1: Consider code C in Example 2.1. Assume that we want to encode $\boldsymbol{v} = 010$. The first step is to encode $(\boldsymbol{v}, 0) = 0100$ into the $[7, 4]$ Hamming code. This gives $\boldsymbol{c} = 0100101$. Since $w(\boldsymbol{c}) = 3$, $\boldsymbol{t}_{w(\boldsymbol{c})} = 00$. The encoded vector is then $\boldsymbol{v} = (\boldsymbol{c}, \boldsymbol{t}_3) = 010010100$.

Similarly, assume that we want to encode $\boldsymbol{u} = 110$. The encoding of $(\boldsymbol{u}, 0) = 1100$ into C' gives $\boldsymbol{c} = 1100110$. Since $w(\boldsymbol{c}) = 4 > 3 = \lfloor n'/2 \rfloor$, then $\boldsymbol{c} = 1111111 \oplus 1100110 = 0011001$. As before, the encoded vector is $\boldsymbol{v} = (\boldsymbol{c}, \boldsymbol{t}_3) = 001100100$.

The decoding is also very simple. Essentially, it works as in [1], with the extra step of taking complements when necessary.

Algorithm 3.2 (Decoding Algorithm) Let C be the EC/AUED obtained from Construction 2.1. Let $\hat{\boldsymbol{v}}$ be the received word and $\hat{\boldsymbol{c}}$ the first n' bits of $\hat{\boldsymbol{v}}$. Then:

1) Decode $\hat{\boldsymbol{c}}$ with respect to C'. If more than t errors, declare an uncorrectable error. Else let $\boldsymbol{c}$ be the corrected word.
2) Let $\boldsymbol{v} = (\boldsymbol{c}, \boldsymbol{t}_{w(\boldsymbol{c})})$. If $d_H(\hat{\boldsymbol{v}}, \boldsymbol{v}) > t$ (d_H denotes Hamming distance), then declare an uncorrectable error.
3) Else, let $u_1, u_2, \cdots, u_{k+1}$ be the $k+1$ information bits from codeword $\boldsymbol{c} \in C'$. Then, the output of the decoder is given by the vector of length k

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

TABLE II
PARAMETERS OF SOME 1-EC/AUED CODES

k	n'	$\lfloor n'/2 \rfloor$	r	$n-k$	$n-k$ from [1]	EC-Code
3	7	3	2	6	7	Hamming
10	15	7	4	9	10	Hamming
22	28	14	6	12	13	Hamming_s
25	31	15	6	12	13	Hamming
87	95	47	8	16	17	Hamming_s
246	255	127	10	19	20	Hamming
277	287	143	10	20	21	Hamming_s

TABLE III
PARAMETERS OF SOME 2-EC/AUED CODES

k	n'	$\lfloor n'/2 \rfloor$	r	$n-k$	$n-k$ from [1]	EC-Code
6	15	7	4	13	15	BCH
15	26	13	7	18	20	BCH_s
20	31	15	7	18	20	BCH
45	58	29	10	23	24	BCH_s
50	63	31	10	23	24	BCH
107	122	61	12	27	28	BCH_s
112	127	63	12	27	28	BCH
222	239	119	13	30	31	BCH_s

Example 3.2: Again consider the code of Examples 2.1 and 3.1.

1) Assume we receive $\hat{\boldsymbol{v}} = 100101110$. According to the Decoding Algorithm, we first consider $\hat{\boldsymbol{c}} = 1001011$. The parity check matrix of C' is
$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$
So, we obtain the syndrome $\boldsymbol{s} = \hat{\boldsymbol{c}}H^T = 111$ which corresponds to the fourth column of H, hence $\hat{\boldsymbol{c}}$ is decoded as $\boldsymbol{c} = 1000011$. Now, $\boldsymbol{v} = (\boldsymbol{c}, \boldsymbol{t}_{w(\boldsymbol{c})}) = 100001100$, hence $d_H(\hat{\boldsymbol{v}}, \boldsymbol{v}) = 2 > 1 = t$. Thus, the decoder declares an uncorrectable error.
2) Assume we receive $\hat{\boldsymbol{v}} = 011011000$. As before, $\hat{\boldsymbol{c}} = 0110110$, and $\boldsymbol{s} = \hat{\boldsymbol{c}}H^T = 101$, which corresponds to the second column of H. Hence, $\hat{\boldsymbol{c}}$ is decoded as $\boldsymbol{c} = 0010110$. So, $\boldsymbol{v} = (\boldsymbol{c}, \boldsymbol{t}_{w(\boldsymbol{c})}) = 001011000$ and $d_H(\hat{\boldsymbol{c}}, \boldsymbol{c}) = 1$. Since $u_4 = 0$, the output of the decoder is $\boldsymbol{u} = 001$.
3) Assume we receive $\hat{\boldsymbol{v}} = 001110100$. Now $\hat{\boldsymbol{c}} = 0011101$, and $\boldsymbol{s} = \hat{\boldsymbol{c}}H^T = 100$, which corresponds to the fifth column of H. Hence $\hat{\boldsymbol{c}}$ is decoded as $\boldsymbol{c} = 0011001$. So, $\boldsymbol{v} = (\boldsymbol{c}, \boldsymbol{t}_{w(\boldsymbol{c})}) = 001100100$ and $d_H(\hat{\boldsymbol{c}}, \boldsymbol{c}) = 1$. Since $u_4 = 1$, the output of the decoder is $\boldsymbol{u} = 001 \oplus 111 = 110$.

IV. TABLES AND COMPARISONS

We have seen in Example 2.1 that we gained one bit with our construction with respect to [1]. In this section, we show that this is not an isolated case.

As stated in Section II, Table I contains the parameters of some descending tail matrices $T(m, r; t+1)$ obtained from [1].

Construction 2.1 ties the results from [1] in most cases, but quite often it also improves them, as shown in Tables II–V.

The tables have seven columns. The first column contains the number of information bits k. The second column gives the length n' of the EC-code. Column 3 contains $\lfloor n'/2 \rfloor$. Column 4 gives the number of extra bits r that we have to add to the EC-code in order to obtain a t-EC/AUED code (Construction 2.1). Column 5 gives the total redundancy $n - k = n' - k + r$ used in the Construction. Column 6 gives the total redundancy obtained using the codes in [1]. Finally, column 7 indicates the EC-code used (containing the all-1 vector). The subscript "s" indicates a shortened code.

TABLE IV
PARAMETERS OF SOME 3-EC/AUED CODES

k	n'	$\lfloor n'/2 \rfloor$	r	$n-k$	$n-k$ from [1]	EC-Code
4	15	7	4	15	18	BCH
11	23	11	6	18	21	Golay
15	31	15	8	24	26	BCH
37	56	28	12	31	32	BCH_s
44	63	31	12	31	32	BCH
105	127	63	15	37	38	BCH
214	239	119	16	41	42	BCH_s
483	511	255	18	46	47	BCH

TABLE V
PARAMETERS OF SOME 4-EC/AUED CODES

k	n'	$\lfloor n'/2 \rfloor$	r	$n-k$	$n-k$ from [1]	EC-Code
38	63	31	13	38	42	BCH
98	127	63	18	47	48	BCH
222	255	127	20	53	54	BCH

Notice that we use only BCH codes and the Golay code, which are easy to decode, while in [1] the best codes of [11] have been chosen. Sometimes no efficient decoder is known for the best possible code.

In order to shorten a code containing the all-1 vector in such a way that the shortened code also contains the all-1 vector, we use the following lemma.

Lemma 4.1: Let $\mathcal{C}$ be an $[n, k, d]$ EC code with parity-check matrix H. Assume that the all-1 vector is in $\mathcal{C}$. Let $\boldsymbol{c}$ be a codeword in $\mathcal{C}$ such that its nonzero components are $i_1, i_2, \cdots, i_w$, $1 \le i_1 < i_2 < \cdots < i_w \le n$. Let $\tilde{H}$ be the matrix obtained by deleting columns $i_1, i_2, \cdots, i_2$ from H. Let $\tilde{\mathcal{C}}$ be the $[n-w, k-w, d]$ code whose parity check matrix is $\tilde{H}$. Then the all-1 vector is in $\tilde{\mathcal{C}}$.

Proof: The all-1 vector is in $\tilde{\mathcal{C}}$ if and only if the sum (modulo 2) of all the columns in $\tilde{H}$ gives the zero column.

Since the all-1 vector 1 is in $\mathcal{C}$, then $1 \oplus \boldsymbol{c}$ is also in $\mathcal{C}$. This vector has zero components $i_1, i_2, \cdots, i_w$. Summing the columns corresponding to the nonzero components, we obtain the zero column. But these columns correspond to the columns in $\tilde{H}$. □

Example 4.1: Consider the [7, 4] Hamming code of Example 2.1. Take codeword $\boldsymbol{c} = 1110000$. In order to obtain matrix $\tilde{H}$ according to Lemma 4.1, we have to delete the first three columns of matrix H of Example 3.2. This gives
$$\tilde{H} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$
The shortened Hamming code has length 4 and dimension 1. The all-1 vector is in the shortened code.

We use this procedure to shorten several of the codes presented in Tables II–V.

V. IMPROVEMENTS ON THE TAIL MATRIX

In this section we show some methods to improve upon the parameters given in Table I. Basically, fixing r and t, we want to obtain a descending tail matrix $T(m, r; t+1)$ such that m is larger

TABLE VI
PARAMETERS OF IMPROVED DESCENDING TAIL MATRICES $T(m, r; t+1)$

t	r	m	t	r	m	t	r	m	t	r	m
1	2	4	2	3	6	3	4	8	4	5	10
1	3	6	2	4	8	3	5	10	4	6	12
1	4	$9^{(5)}$	2	5	10	3	6	12	4	7	14
1	5	12	2	6	12	3	7	14	4	8	16
1	6	$18^{(5)}$	2	7	16	3	8	16	4	9	18
1	7	$29^{(6)}$	2	8	20	3	9	20	4	10	20
1	8	$50^{(5)}$	2	9	24	3	10	$26^{(2)}$	4	11	24
1	9	$74^{(5)}$	2	10	32	3	11	28	4	12	28
1	10	$146^{(5)}$	2	11	$52^{(1)}$	3	12	32	4	13	32
1	11	$250^{(5)}$	2	12	$76^{(1)}$	3	13	40	4	14	36
1	12	$434^{(5)}$	2	13	$124^{(1)}$	3	14	$56^{(1)}$	4	15	40
			2	14	$220^{(1)}$	3	15	$80^{(1)}$	4	16	48
			2	15	$396^{(1)}$	3	16	$128^{(1)}$	4	17	$60^{(1)}$
						3	17	$184^{(1)}$	4	18	$80^{(3)}$
						3	18	$272^{(1)}$	4	19	$116^{(1)}$
						3	19	$496^{(1)}$	4	20	$164^{(3)}$
									4	21	$224^{(3)}$
									4	22	$292^{(4)}$

1. From Construction 5.1
2. From Example 5.1
3. From Construction 5.2
4. From Construction 5.3
5. From Construction 5.4
6. From Example 5.2

than the one given in Table I. The parameters in Table I come from [1]. Table VI improves on Table I based on the results of this section. The superindexes in Table VI denote the entries from Table I that have been improved and the method (to be described later in this section) used in the improvement.

Let us recall briefly how the descending tail matrices were obtained in [1].

Let A be an $m \times r$ matrix with rows $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m$ and B an $m' \times r'$ matrix with rows $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_{m'}$. Then, we denote by $A \times B$ (also called external or Kronecker product) the $(mm') \times (r + r')$ matrix whose $(i-1)m' + j$ row, $1 \le i \le m, 1 \le j \le m'$ is vector $\boldsymbol{a}_i, \boldsymbol{b}_j$. Given $j \ge 1$, let T_j be the matrix constructed inductively as follows:

$$T_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad T_{j+1} = \left(\begin{array}{ccc|c} 1 & 1 & \ldots & 1 \\ \hline & & & 0 \\ & & & 1 \\ & & T_j & \vdots \\ & & & 0 \\ & & & 1 \\ \hline 0 & 0 & \ldots & 0 \end{array}\right).$$

For example,

$$T_2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad T_3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Given r and t, let $\mathcal{A}$ be an asymmetric t-error correcting code of length $r' < r$ and cardinality $|\mathcal{A}| = m'$. Let $A(m', r'; t+1)$ be the matrix obtained by ordering the elements of $\mathcal{A}$ in descending weight order. Then, the matrix $A(m', r'; t+1) \times T_{r-r'}$ is a descending tail matrix $T(m, r; t+1)$, where $m = 2(r - r')m'$ [1]. We call a matrix of this type a Blaum-van Tilborg (BT) descending tail matrix of strength $t+1$. The construction is optimized over the best asymmetric t-error correcting codes available and the matrices T_j in order to obtain m as large as possible. Results concerning good asymmetric error correcting codes can be found in [12].

Example 5.1: Let $\mathcal{A}$ be the 3-asymmetric error correcting code of length 8. The rows of the corresponding matrix $A(4, 8; 4)$ are

$$\boldsymbol{a}_1 = 11111111$$
$$\boldsymbol{a}_2 = 00001111$$
$$\boldsymbol{a}_3 = 11110000$$
$$\boldsymbol{a}_4 = 00000000.$$

The BT matrix $T(16, 10; 4)$ is then

$$A(4,8;4) \times T_2 = \begin{array}{|c|c|}\hline 11111111 & 11 \\ & 10 \\ & 01 \\ & 00 \\ \hline 00001111 & 11 \\ & 10 \\ & 01 \\ & 00 \\ \hline 11110000 & 11 \\ & 10 \\ & 01 \\ & 00 \\ \hline 00000000 & 11 \\ & 10 \\ & 01 \\ & 00 \\ \hline \end{array} = \begin{array}{|l|}\hline 11111111 \times T_2 \\ \hline 00001111 \times T_2 \\ \hline 11110000 \times T_2 \\ \hline 00000000 \times T_2 \\ \hline \end{array}$$

We show next how we can construct a descending tail matrix T by inserting some rows in $A(4, 8; 4) \times T_2$. The new matrix is

$$T = \begin{array}{|l|}\hline 11111111 \times T_2 \\ \hline 01111111\ 01 \\ 00111111\ 10 \\ 00111111\ 01 \\ 00011111\ 10 \\ \hline 00001111 \times T_2 \\ \hline 11001100\ 10 \\ 11001100\ 01 \\ \hline 11110000 \times T_2 \\ \hline 01110000\ 01 \\ 00110000\ 10 \\ 00110000\ 01 \\ 00010000\ 10 \\ \hline 00000000 \times T_2 \\ \hline \end{array}$$

It is easy to verify that the new matrix is a $T(26, 10; 4)$ descending tail matrix. In Table I, we can see that our best result was $m = 24$. This example shows the potential of the method. In the discussion that follows, whenever we have a matrix $A(m, r'; t+1)$ with rows $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m$, we make the following assumption: $\boldsymbol{a}_1 = 11\cdots1, \boldsymbol{a}_2 = \overbrace{00\cdots0}^{t+1}11\cdots1, a_{m-1} = \overbrace{11\cdots1}^{t+1}00\cdots0$ and $\boldsymbol{a}_m = 00\cdots0$. This is not entirely correct, though. We can make the assumption on $\boldsymbol{a}_2$ and on $\boldsymbol{a}_{m-1}$ separately by an adequate permutation of the columns of $A(m, r'; t+1)$. In general, there is not necessarily a permutation that can take $\boldsymbol{a}_2$ and $\boldsymbol{a}_{m-1}$ to the form described above simultaneously. However, this fact is not very relevant in the construction that follows. The real assumption is $\boldsymbol{a}_2 = \overbrace{00\cdots0}^{t+1}11\cdots1$, and a_{m-1} is a permutation of $\overbrace{11\cdots1}^{t+1}00\cdots0$.

The next construction partially generalizes Example 5.1.

Construction 5.1: Let $A(m, r-2; t+1) \times T_2$ be a BT descending tail matrix $T(4m, r; t+1)$, where $t \geq 2$ and $m \geq 3$. Denote the rows of $A(m, r-2; t+1)$ $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_m$. Let B and B' be the following $T(2(t-1), r; t+1)$ descending tail matrices:

$$B = \begin{array}{|c|c|}\hline \overbrace{01111\ldots1}^{t+1}11\ldots1 & 01 \\ \overbrace{00111\ldots1}^{t+1}11\ldots1 & 10 \\ \overbrace{00111\ldots1}^{t+1}11\ldots1 & 01 \\ \overbrace{00011\ldots1}^{t+1}11\ldots1 & 10 \\ \overbrace{00011\ldots1}^{t+1}11\ldots1 & 01 \\ \ldots & \ldots \\ \overbrace{00\ldots011}^{t+1}11\ldots1 & 10 \\ \overbrace{00\ldots011}^{t+1}11\ldots1 & 01 \\ \overbrace{00\ldots001}^{t+1}11\ldots1 & 10 \\ \hline \end{array} \quad \text{and} \quad B' = \begin{array}{|c|c|}\hline \overbrace{01111\ldots1}^{t+1}00\ldots0 & 01 \\ \overbrace{00111\ldots1}^{t+1}00\ldots0 & 10 \\ \overbrace{00111\ldots1}^{t+1}00\ldots0 & 01 \\ \overbrace{00011\ldots1}^{t+1}00\ldots0 & 10 \\ \overbrace{00011\ldots1}^{t+1}00\ldots0 & 01 \\ \ldots & \ldots \\ \overbrace{00\ldots011}^{t+1}00\ldots0 & 10 \\ \overbrace{00\ldots011}^{t+1}00\ldots0 & 01 \\ \overbrace{00\ldots001}^{t+1}00\ldots0 & 10 \\ \hline \end{array}$$

Then, the following matrix T is a $T(4m + 4(t-1), r; t+1)$ descending tail matrix:

$$T = \begin{array}{|c|}\hline \boldsymbol{a}_1 \times T_2 \\ \hline B \\ \hline \boldsymbol{a}_2 \times T_2 \\ \boldsymbol{a}_3 \times T_2 \\ \cdots \\ \boldsymbol{a}_{m-1} \times T_2 \\ \hline B' \\ \hline \boldsymbol{a}_m \times T_2 \\ \hline \end{array}$$

We prove that matrix T above is a $T(4m + 4(t-1), r; t+1)$ descending tail matrix in the Appendix.

Example 5.2: Applying Construction 5.1 to the $T(16, 10, ; 4)$ BT matrix $A(4, 8; 4) \times T_2$ in Example 5.1, we obtain

$$T' = \begin{array}{|l|}\hline 11111111 \times T_2 \\ \hline 01111111\ 01 \\ 00111111\ 10 \\ 00111111\ 01 \\ 00011111\ 10 \\ \hline 00001111 \times T_2 \\ \hline 11110000 \times T_2 \\ \hline 01110000\ 01 \\ 00110000\ 10 \\ 00110000\ 01 \\ 00010000\ 10 \\ \hline 00000000 \times T_2 \\ \hline \end{array}$$

Matrix T' is a $T(24, 10, 4)$ descending tail matrix. In Example 5.1, we had obtained a $T(26, 10; 4)$ descending tail matrix T. Construction 5.1 only shows how to insert rows between blocks 1 and 2 and between blocks $m-1$ and m. However, in individual cases, depending on the asymmetric code used, it is possible to insert rows between the middle blocks, as shown in Example 5.1. Table VI shows improvements over Table I using Construction 5.1 together with other constructions to be presented in this section. However, the reader can improve further on Table VI by taking individual cases and inserting rows between the middle blocks.

A BT descending tail matrix $A(m, r'; t+1) \times T_j$ can be considered as a sequence of m blocks $\boldsymbol{a}_1 \times T_j, \boldsymbol{a}_2 \times T_j, \cdots, \boldsymbol{a}_m \times T_j$. Construction 5.1 gives a way of inserting extra rows between blocks 1 and 2 and between blocks $m-1$ and m when $t \geq 2, m \geq 3$ and $j = 2$. In [1], most of the constructions involve $j = 2, j = 3$, and $j = 4$. The next two constructions show how to insert rows between blocks 1 and 2 and between blocks $m-1$ and m when $t = 4$ and $j = 3$ or $j = 4$. The proofs that the new matrices are still descending tail matrices of strength $t+1$ are similar to the case $j = 2$ and will be omitted.

Construction 5.2: Assume that we have a BT $(6m, r; 5)$ descending tail matrix $A(m, r-3; 5) \times T_3 (m \geq 3)$. Let B and B' be the following $4 \times r$ matrices:

$$B = \begin{array}{|l|l|}\hline 011111\ldots1 & 010 \\ 001111\ldots1 & 001 \\ 000111\ldots1 & 110 \\ 000011\ldots1 & 101 \\ \hline \end{array} \quad \text{and } B' = \begin{array}{|l|l|}\hline 0111100\ldots0 & 010 \\ 0011100\ldots0 & 001 \\ 0001100\ldots0 & 110 \\ 0000100\ldots0 & 101 \\ \hline \end{array}$$

Then, the following matrix T is a $T(6m + 8, r; 5)$ descending tail matrix:

$$T = \begin{array}{|c|}\hline \boldsymbol{a}_1 \times T_3 \\ \hline B \\ \hline \boldsymbol{a}_2 \times T_3 \\ \hline \boldsymbol{a}_3 \times T_3 \\ \hline \vdots \\ \hline \boldsymbol{a}_{m-1} \times T_3 \\ \hline B' \\ \hline \boldsymbol{a}_m \times T_3 \\ \hline \end{array}$$

Construction 5.3: Assume that we have a BT $T(8m, r; 5)$ descending tail matrix $A(m, r-4; 5) \times T_4 (m \geq 3)$. Let B and B' be the following $2 \times r$ matrices:

$$B = \begin{array}{|l|l|}\hline 001111\ldots1 & 0011 \\ 000111\ldots1 & 1100 \\ \hline \end{array} \quad \text{and} \quad B' = \begin{array}{|l|l|}\hline 0011100\ldots0 & 0011 \\ 0001100\ldots0 & 1100 \\ \hline \end{array}$$

Then, the following matrix T is a $T(8m + 4, r; 5)$ descending tail matrix:

$$T = \begin{array}{|c|}\hline \boldsymbol{a}_1 \times T_4 \\ \hline B \\ \hline \boldsymbol{a}_2 \times T_4 \\ \hline \boldsymbol{a}_3 \times T_4 \\ \hline \vdots \\ \hline \boldsymbol{a}_{m-1} \times T_4 \\ \hline B' \\ \hline \boldsymbol{a}_m \times T_4 \\ \hline \end{array}$$

Up to now, we have analyzed codes with $t \geq 2$. Let us turn now our attention to the case $t = 1$.

We have that $T_2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$. Similarly, we can define $T_2' = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$.

The next construction shows how to modify a BT $T(4m, r; 2)$ descending tail matrix in such a way that a $T(4m+2, r; 2)$ descending tail matrix is obtained. The proof will be omitted.

Construction 5.4: Consider an $A(m, r-2; 2)$ matrix. Then, if m is even, the following matrix is a $T(4m+2, r; 2)$ descending tail matrix:

$$T = \begin{array}{|c|}\hline \boldsymbol{a}_1 \times T_2 \\ \hline 0111...1\ 01 \\ \hline \boldsymbol{a}_2 \times T_2' \\ \hline \boldsymbol{a}_3 \times T_2 \\ \hline \cdots \\ \hline \boldsymbol{a}_{m-1} \times T_2 \\ \hline 0100...0\ 01 \\ \hline \boldsymbol{a}_m \times T_2' \\ \hline \end{array}$$

If m is odd, the same is true for the following matrix:

$$T = \begin{array}{|c|}\hline \boldsymbol{a}_1 \times T_2 \\ \hline 0111...1\ 01 \\ \hline \boldsymbol{a}_2 \times T_2' \\ \hline \boldsymbol{a}_3 \times T_2 \\ \hline \cdots \\ \hline \boldsymbol{a}_{m-1} \times T_2' \\ \hline 0100...0\ 10 \\ \hline \boldsymbol{a}_m \times T_2 \\ \hline \end{array}$$

Example 5.3: Let $r = 7$ and $t = 2$. Then, the following is a BT $T(24, 7; 2)$ descending tail matrix:

$$A(6, 5; 2) \times T_2 = \begin{array}{|c|}\hline 11111 \times T_2 \\ \hline 11100 \times T_2 \\ \hline 10011 \times T_2 \\ \hline 00101 \times T_2 \\ \hline 01010 \times T_2 \\ \hline 00000 \times T_2 \\ \hline \end{array}$$

According to Construction 5.4, the following is a $T(26, 7; 2)$ descending tail matrix:

$$T = \begin{array}{|c|}\hline 11111 \times T_2 \\ \hline 11110\ 01 \\ \hline 11100 \times T_2' \\ \hline 10011 \times T_2 \\ \hline 00101 \times T_2' \\ \hline 01010 \times T_2 \\ \hline 01000\ 01 \\ \hline 00000 \times T_2' \\ \hline \end{array}$$

However, we can improve further and obtain a $T(29, 7; 2)$ descending tail matrix by adding more rows as follows:

$$T = \begin{array}{|c|}\hline 11111 \times T_2 \\ \hline 11110\ 01 \\ \hline 11100 \times T_2' \\ \hline 11001\ 10 \\ \hline 10011 \times T_2 \\ \hline 00111\ 01 \\ \hline 00101 \times T_2' \\ \hline 00110\ 10 \\ \hline 01010 \times T_2 \\ \hline 01000\ 01 \\ \hline 00000 \times T_2' \\ \hline \end{array}$$

It is straightforward to verify that T' is indeed a $T(29, 7; 2)$ descending tail matrix.

As Example 5.3 shows, we can improve on Table VI by taking separately each individual case and inserting more rows in an ad hoc way that depends on the asymmetric error correcting code considered.

VI. Conclusions

Some new methods for constructing t-EC/AUED codes have been presented. In the first method, the information bits are encoded first into a t-EC code containing the all-1 vector. Since most codes used in applications have this property (BCH, shortened BCH, Golay), this is a natural assumption. The key idea in the construction is reducing the weight distribution of the t-EC code used. Our codes have frequently less redundancy than the best EC/AUED codes previously known. The encoding and decoding procedures are as simple as those of known codes.

The second procedure improves the tail matrices obtained by a previously known method. The results of the new method, together with some ad hoc constructions given in examples, are tabulated in Table VI. Table VI should not be taken as a new record on m. The reader should be able to improve upon Table VI by ad hoc methods similar to the ones given in the examples.

Appendix

Lemma: The matrix T obtained in Construction 5.1 is a $T(4m + 4(t-1), r; t+1)$ descending tail matrix.

Proof: We have to prove that the rows corresponding to B and B' in T comply with Definition 1.1. Denote the rows of T as $\boldsymbol{t}_j, 1 \le j \le 4m + 4(t-1)$. Notice that the rows corresponding to B are rows t_j, $5 \le j \le 2(t+1)$ in T, while the rows corresponding to B' are rows t_j, $4(m-1)+2(t-1)+1 \le j \le 4(m-1)+4(t-1)$ in T. Take the first $r-2$ elements of any row of B. We obtain a vector of length $r-2$ that contains $\boldsymbol{a}_2$, denote it by v. Since $N(\boldsymbol{a}_2, \boldsymbol{a}_j) \ge t+1$ for all $j \ge 3$, also $N(\boldsymbol{v}, \boldsymbol{a}_j) \ge t+1$. Now take the first $r-2$ elements of any row in B', call the resulting vector $\boldsymbol{v}'$. Since $\boldsymbol{v}'$ is contained in $\boldsymbol{a}_{m-1}$, it is also clear that $N(\boldsymbol{v}, \boldsymbol{v}') \ge t+1$. So, it is enough to compare the rows of B with the rows of $\boldsymbol{a}_1 \times T_2$ and of $\boldsymbol{a}_2 \times T_2$. Similarly, it is enough to compare the rows of B' with the rows of $\boldsymbol{a}_{m-1} \times T_2$ and of $\boldsymbol{a}_m \times T_2$.

Take the first element of B in T, i.e., $\boldsymbol{t}_5 = 011...1\ 01$. We have the following:

$$N(\boldsymbol{t}_1, \boldsymbol{t}_5) = N(11...1\ 11,\ 011...1\ 01) = 2$$

$$N(\boldsymbol{t}_2, \boldsymbol{t}_5) = N(11...1\ 10,\ 011...1\ 01) = 2$$

$$N(\boldsymbol{t}_3, \boldsymbol{t}_5) = N(11...1\ 01,\ 011...1\ 01) = 1$$

$$N(\boldsymbol{t}_4, \boldsymbol{t}_5) = N(11...1\ 00,\ 011...1\ 01) = 1.$$

So, Definition 1.1 is satisfied for t_j, $1 \leq j \leq 4$ with respect to t_5. Also, notice that

$$N(t_5, t_{2(t+1)+1}) = N\left(011...1\ 01,\ \overbrace{00...0}^{t+1}11...1\ 11\right) = t$$

$$N(t_5, t_{2(t+1)+2}) = N\left(011...1\ 01,\ \overbrace{00...0}^{t+1}11...1\ 10\right) = t+1$$

$$N(t_5, t_{2(t+1)+3}) = N\left(011...1\ 01,\ \overbrace{00...0}^{t+1}11...1\ 01\right) = t$$

$$N(t_5, t_{2(t+1)+4}) = N\left(011...1\ 01,\ \overbrace{00...0}^{t+1}11...1\ 00\right) = t+1.$$

So, Definition 1.1 is satisfied for t_j, $2(t+1)+1 \leq j \leq 2(t+1)+4$ with respect to t_5. Similarly, we verify Definition 1.1 with respect to $t_{2(t+1)}$.

Let us consider now the rows t_j, $6 \leq j \leq 2t-1$. Take $t_{6+2l} = \overbrace{00...0}^{l+2}11...1\ 10$, $0 \leq l \leq t-3$. Notice that

$$N(t_1, t_{6+2l}) = N\left(11...1\ 11, \overbrace{00...0}^{l+2}11...1\ 10\right) = l+3$$

$$N(t_2, t_{6+2l}) = N\left(11...1\ 10, \overbrace{00...0}^{l+2}11...1\ 10\right) = l+2$$

$$N(t_3, t_{6+2l}) = N\left(11...1\ 01, \overbrace{00...0}^{l+2}11...1\ 10\right) = l+3$$

$$N(t_4, t_{6+2l}) = N\left(11...1\ 00, \overbrace{00...0}^{l+2}11...1\ 10\right) = l+2$$

satisfying Definition 1.1 on block $a_1 \times T_2$ with respect to t_{6+2l}. Similarly,

$$N(t_{6+2l}, t_{2(t+1)+1}) = N(\overbrace{00...0}^{l+2}11...1\ 10,\ \overbrace{00...0}^{t+1}11...1\ 11) = t-l-1$$

$$N(t_{6+2l}, t_{2(t+1)+2}) = N(\overbrace{00...0}^{l+2}11...1\ 10,\ \overbrace{00...0}^{t+1}11...1\ 10) = t-l-1$$

$$N(t_{6+2l}, t_{2(t+1)+3}) = N(\overbrace{00...0}^{l+2}11...1\ 10,\ \overbrace{00...0}^{t+1}11...1\ 01) = t-l$$

$$N(t_{6+2l}, t_{2(t+1)+4}) = N(\overbrace{00...0}^{l+2}11...1\ 10,\ \overbrace{00...0}^{t+1}11...1\ 00) = t-l$$

showing that Definition 1.1 is satisfied on block $a_2 \times T_2$ with respect to t_{6+2l}. We show analogously that Definition 1.1 is satisfied with respect to elements t_{6+2l+1}.

Similarly, we verify Definition 1.1 on blocks $a_{m-1} \times T_2$ and $a_m \times T_2$ with respect to the rows of B', completing the proof.

References

[1] M. Blaum and H. van Tilborg, "On t-error correcting/all unidirectional error detecting codes," *IEEE Trans. Comput.*, vol. 38, no. 11, pp. 1493–1501, Nov. 1989.

[2] F. J. H. Boinck and H. van Tilborg, "Constructions and bounds for systematic t EC/AUED codes," *IEEE Trans. Inform. Theory*, vol. IT-36, pp. 1381–1390, Nov. 1990.

[3] B. Bose, "On systematic SEC/MUED codes," in *Proc. FTCS*, vol. 11, June 1981, pp. 265–267.

[4] B. Bose and D. K. Pradhan, "Optimal unidirectional error detecting/correcting codes," *IEEE Trans. Comput.*, vol. C-32, pp. 521–530, June 1982.

[5] B. Bose and T. R. N. Rao, "On the theory of unidirectional error correcting/detecting codes," *IEEE Trans. Comput.*, vol. C-31, pp. 521–530, June 1982.

[6] S. Kundu, "Design of testable CMOS circuits for TSC systems," Ph.D. dissertation, Univ. of Iowa, May 1988.

[7] D. Nikolos, N. Gaitanis, and G. Philokyprou, "t-error correcting all unidirectional error detecting codes starting from cyclic AN codes," in *Proc. Int. Conf. Fault-Tolerant Comput.*, Kissimmee, FL 1984, pp. 318–323.

[8] ——, "Systematic t-error correcting/all unidirectional error detecting codes," *IEEE Trans. Comput.*, vol. C-35, pp. 394–402, May 1986.

[9] D. K. Pradhan, "A new class of error-corrrecting/detecting codes for fault-tolerant computer applications," *IEEE Trans. Comput.*, vol. C-29, pp. 471–481, June 1980.

[10] D. L. Tao, C. R. P. Hartmann, and P. K. Lala, "An efficient class of unidirectional error detecting/correcting codes," *IEEE Trans. Comput.*, vol. C-37, pp. 879–882, July 1988.

[11] T. Verhoeff, "An updated table of minimum-distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 665–680, Sept. 1987.

[12] J. H. Weber, C. de Vroedt, and D. E. Boekee, "Bounds and constructions for binary codes of length less than 24 and asymmetric distance less than 6," *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1321–1331, Sept. 1988.

Adaptation of the Mactaggart and Jack Complex Multiplication Algorithm for Floating-Point Operators

Ronald J. Cosentino and John J. Vaccaro

***Abstract*— With a suitable treatment of the exponents in the input operands, a hardware implementation of the Mactaggart and Jack fixed-point complex multiplication algorithm can also calculate a floating-point product with no loss in accuracy from the greater dynamic range of the floating-point inputs. This floating-point technique can be extended to any sum of two products operation, such as encountered in matrix multiplication and vector cross-products.**

***Index Terms*— Complex multiplication, distributed arithmetic, FFT butterfly, floating-point multiplication.**

I. R. Mactaggart and M. A. Jack have developed an algorithm and hardware implementation of an FFT butterfly using fixed-point multiplication of complex numbers that requires, in effect, two real multiplications instead of the conventional four real multiplications [1]. This saving allows an implementation of the Mactaggart and Jack algorithm to be more area-efficient than the modified Booth

Manuscript received April 5, 1990; revised December 15, 1990. This work was supported by the Electronic Systems Division, U.S.A.F., Hanscom Air Force Base, MA.

The authors are with The MITRE Corporation, Bedford, MA 01730.

IEEE Log Number 9103105.