New Encoding/Decoding Methods for Designing Fault-Tolerant Matrix Operations

D.L. Tao, C.R.P. Hartmann, and Yunghsing S. (Sam) Han

Abstract—Algorithm-based fault tolerance (ABFT) can provide a low-cost error protection for array processors and multiprocessor systems. Several ABFT techniques (*weighted check-sum*) have been proposed to design fault-tolerant matrix operations. In these schemes, encoding/decoding uses either multiplications or divisions so that overhead is high. In this paper, new encoding/decoding methods are proposed for designing fault-tolerant matrix operations. The unique feature of these new methods is that only additions and subtractions are used in encoding/decoding. In this paper, new algorithms are proposed to construct error detecting/correcting codes with the minimum Hamming distance 3 and 4. We will show that the overhead introduced due to the incorporation of fault tolerance is drastically reduced by using these new coding schemes.

Index Terms—Array processors, concurrent error detection/correction, error detecting/correcting codes, fault tolerance, multiprocessor systems.

1 Introduction

ALGORITHM-BASED fault tolerance (ABFT) has been suggested as a means for designing fault tolerant array processors and multiprocessor systems. The advantage of ABFT is that errors which are caused by permanent or transient failures in the system can be detected/corrected by using a very low overhead and at the original throughput. ABFT was originated by Huang and Abraham [7]. In [7], a checksum approach is proposed to provide concurrent error detection/correction for matrix operations. The technique has been extended to many digital signal processing (DSP) applications, such as matrix computations [1], [4], [8], [10], [13], FFT [9], [17], [18], matrix equation solvers [11] eigenvalue, and singular value problems [5]. Moreover, the idea of ABFT is extended to design fault-tolerant multiprocessor systems [2], [3], [6], [15], [20].

Existing ABFT techniques use various coding schemes to provide low-cost error protection for processor arrays, and they have the block diagram as shown in Fig. 1. Note that these coding schemes cannot be utilized to provide fault tolerance for other parts of a computer system, e.g., main memory and I/O devices. As a result, encoding/decoding must be considered as overhead introduced by ABFT. In [8], [10], [13], multiplication and division operations are used in encoding/decoding, and so the complexity of overhead is very high.

In this paper, we introduce a new class of encoding/decoding techniques for designing ABFT. The resulting ABFT can be used to perform concurrent error detection/correction on a class of matrix operations which include matrix-vector multiplication, matrix-matrix multiplication, LU decomposition, QR decomposition, and matrix inversion. During system normal operations, the probability of having more than two faults is negligible. Hence, two new encoding/decoding algorithms to construct error detecting/correcting codes with the minimum Hamming distance 3 and 4 are respectively proposed in this paper. We will show that overhead of the proposed schemes (in terms of additional operations and in terms of hardware complexity) is significantly less than that used in [8], [10], [13]. In ABFT, a fault manifests as a single error. Hence, when the new scheme with the minimum distance 3 (4) is mapped into an array processor to compute fault-tolerant matrix operations, all single faults can be tolerated (all single faults can be tolerated and all double faults will be simultaneously detected).

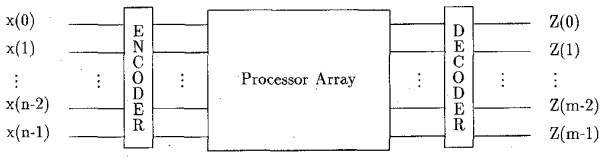


Fig. 1. The block diagram of algorithm-based fault tolerance (ABFT) techniques.

The fault model used in this paper is the same as that used in [7], [8], i.e., a fault is modeled as a faulty PE which may produce any arbitrary error under failures. A faulty PE will manifest as a single data error because the computation steps in the algorithms have been distributed among many PEs in ABFT. Hence, for the sake of simplicity, we use the words "fault" and "error" interchangeably in the rest of the paper. The rest of the paper is organized as follows. In Section 2, we discuss several existing encoding/decoding methods. New encoding/decoding schemes are proposed in Section 3. In Section 4, we compare the proposed

- *D.L. Tao is with the Department of Electrical Engineering, State University of New York at Stony Brook, Stony Brook, NY 11794. E-mail: dali@sbee.sunysb.edu.*
- *C.R.P. Hartmann is with the School of Computer and Information Science, Syracuse University, Syracuse, NY 13244.*
- *Y.S. Han is with the Department of Electronics Engineering, HuaFan College of Humanities and Technology, Taipei, Taiwan.*

Manuscript received Mar. 10, 1994.
For information on obtaining reprints of this article, please send e-mail to: transpds@computer.org, and reference IEEECS Log Number D95221.

schemes with other existing schemes in terms of overhead introduced by incorporating fault tolerance. The conclusion is given in Section 5.

2 A Brief Review

In this section, we evaluate existing ABFT techniques [7], [8], [13] which are used to perform fault tolerant matrix-vector multiplications. Note that the advantages and drawbacks of these techniques [7], [8], [13] in matrix-vector multiplication are similar to that in matrix-matrix multiplication, LU decomposition, QR decomposition, and matrix-inversion.

The check-sum scheme proposed in [7] can be used for matrix-vector multiplication, but the overhead of this approach exceeds 100%. This scheme can only detect but not correct errors that occur in LU decomposition and matrix inversion. A *weighted check-sum* scheme is proposed in [8] for detecting/correcting errors that occur in matrix-vector multiplication, LU decomposition, matrix inversion, etc. The following check matrix, H with dimension $(d_{min} - 1) \times (d_{min} - 1 + k)$, is used to construct a code with the minimum distance d_{min}:

$$H = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots w_{1,k} & -1 & 0 & \cdots & 0 \\ w_{2,1} & w_{2,2} & \cdots w_{2,k} & 0 & -1 & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & \cdot & \cdots & \cdot \\ w_{d,1} & w_{d,2} & \cdots w_{d,k} & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Since the probability of having more than two errors is negligible, we first consider single error correction, and then consider single error correction and all double error detection.

2.1 Single Error Correction

For single error correction, i.e., the minimum Hamming distance is 3, the check matrix given in [8] has the following form:

$$H = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots w_{1,k} & -1 & 0 \\ w_{2,1} & w_{2,2} & \cdots w_{2,k} & 0 & -1 \end{bmatrix}.$$

Jou and Abraham have proposed the weight $w_{2,i}$ as 2^i because the multiplication of a number by 2^i can be realized by using a shift operation in a fixed-point system. Hence, the H has the following form:

$$H = \begin{bmatrix} 1 & 1 & \cdots & 1 & -1 & 0 \\ 2^0 & 2^1 & \cdots & 2^{k-1} & 0 & -1 \end{bmatrix}.$$

The weights selected in [8], i.e., 2^0, 2^1, $\cdots$, and 2^{k-1}, are extremely large and can easily cause an overflow in the system. When an overflow occurs, the system performance will be degraded. If an overflow is avoided by using a longer word length in redundant processors, then the redundant processors which are used to compute

$$\sum_{j=1}^{m}\left[\left(\sum_{i=1}^{k} 2^{i-1} \times a_{i,j}\right) \times b_j\right]$$

will be much more complex than the PE to compute

$$\sum_{j=1}^{m}\left(a_{i,j} \times b_j\right).$$

To solve such a problem, three alternative solutions have been proposed:

1) **Jou-Abraham's method with an overflow consideration.** In a fixed-point system, Jou and Abraham propose to use residue arithmetic [16]. Hence, the sums

$$\sum_{i=1}^{k} a_{i,j}$$

and

$$\sum_{i=1}^{k} 2^{i-1} a_{i,j}$$

are replaced by

$$\left[\sum_{i=1}^{k} a_{i,j}\right]_{modM_1}$$

and

$$\left[\sum_{i=1}^{k}\left(2^{i-1} a_{i,j}\right) modM_2\right]_{modM_2},$$

respectively. Thus, two divisors, M_1 and M_2, are required. The divisor M_2 suggested in [8] is the largest possible prime that is less than 2^{l+1}, where l is the word length. For example, if $l = 16$, then $2^{l+1} = 131{,}072$, and the divisor M_2 should be equal to 131,059. To obtain a residue,

$$\left(2^{i-1} a_{i,j}\right)_{modM_2},$$

a division operation is used. As a result, $(k + 1)$ division operations are required to obtain

$$\left[\sum_{i=1}^{k}\left(2^{i-1} a_{i,j}\right) modM_2\right]_{modM_2},$$

and $(k + 1) \times k$ divisions are used in encoding. Similarly, $(k + 1)$ division operations are used in decoding.

2) **Luk and Park's scheme.** Luk and Park alleviate the problem by selecting $w_{2,j}$ as j [13]. As a result, the check matrix H in [13] is given as follows:

$$H = \begin{bmatrix} 1 & 1 & \cdots & 1 & -1 & 0 \\ 1 & 2 & \cdots & k & 0 & -1 \end{bmatrix}.$$

Although this method results in the size of the check-sum increasing less rapidly, an overflow may still occur. To obtain

$$\sum_{i=1}^{k}\left(i \times a_{i,j}\right),$$

k multiplication operations are required, and so k^2 and k multiplication operations are respectively used in encoding and decoding.

3) **A similar scheme is proposed in [10].** The check matrix proposed in [10] has the following form:

$$H = \begin{bmatrix} W_k^0 & W_k^1 & \cdots & W_k^{k-1} & -1 & 0 \\ W_k^0 & W_k^{2\times 1} & \cdots & W_k^{2\times(k-1)} & 0 & -1 \end{bmatrix}$$

where

$$W_k^1 = \exp\left(j\frac{2\pi}{k}\right).$$

This scheme requires even more overhead than Luk and Park's scheme because it requires complex multiplications in encoding/decoding.

The complexity of the above encoding/decoding techniques is summarized in Table 1. It can be seen that the overhead of encoding/decoding in these methods is higher than that of matrix-vector multiplication (a matrix-vector multiplication needs k^2 multiplications and $k \times (k-1)$ additions).

TABLE 1
ADDITIONAL OPERATIONS USED IN ENCODING/DECODING OF EXISTING TECHNIQUES

	JA [8]	LP [13]	KW [10]
$d_{min} = 3$	$(k+2)k$ divisions $2k^2$ additions	$k^2 + k$ multiplications $2k^2$ additions	$2k^2 + 2k$ multiplications $4k^2$ additions
$d_{min} = 4$	$2(k+2)k$ divisions $3k^2$ additions	$2k^2 + 2k$ multiplications $3k^2$ additions	$4k^2 + 4k$ multiplications $6k^2$ additions

2.2 Single Error Correction/Double Error Detection

For single error correction and all double error detection, a minimum Hamming distance 4 code is needed. Hence, the check matrix has the following form.

$$H = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots w_{1,k} & -1 & 0 & 0 \\ w_{2,1} & w_{2,2} & \cdots w_{2,k} & 0 & -1 & 0 \\ w_{3,1} & w_{3,2} & \cdots w_{3,k} & 0 & 0 & -1 \end{bmatrix}.$$

The overhead of encoding/decoding using [8], [13], [10] can be computed in a similar way as given in the previous subsection and is also shown in Table 1.

3 Partial Check-Sum: New Encoding/Decoding Schemes

In this section, we propose new encoding/decoding methods to compute matrix-vector multiplication, matrix-matrix multiplication, LU decomposition, QR decomposition, and matrix inversion. Similar to the schemes given in [8], [10], [13], the encoding/decoding methods proposed in this paper can be described by a check matrix $H^* = [h_{ij}]$. As discussed in Section 2, multiplication/division operations should be avoided in encoding/decoding, and thus the new methods use only additions/subtractions so that the total overhead will be reduced significantly. Two different encoding/decoding methods are discussed in the following two subsections.

3.1 Lengthened Hamming Codes (LHC)

In this subsection, we introduce the *lengthened Hamming code* (LHC) with the minimum Hamming distance 3. So it can correct all single errors. The *lengthened Hamming code* can be constructed by the check matrix $\mathbf{H}^*_{sec}$. Since only additions/subtractions are used in encoding/decoding, the entries in the first k columns of H^*_{sec} can be 0, 1, and –1. Before introducing a procedure to generate an $\mathbf{H}^*_{sec}$, a lemma is introduced to provide a sufficient condition for constructing the $\mathbf{H}^*_{sec}$.

LEMMA. *Let c_i and c_j be two distinct columns in $\mathbf{H}^*_{sec}$, a_1, and a_2 be nonzero real numbers. If $a_1c_i = a_2c_j$, then $c_i = c_j$ or $c_i = -c_j$.*

PROOF. If $a_1c_i = a_2c_j$, then $c_i = \frac{a_2}{a_1}c_j$. Since the entries in c_i and c_j can only be 1, 0, and –1, If $a_1c_i = a_2c_j$, then $\frac{a_2}{a_1}$ can only be 1 or –1. Therefore, $c_i = c_j$ or $c_i = -c_j$. □

For constructing a minimum distance 3 code, every linear combination of two distinct columns of H^*_{sec} must be linearly independent. This requires that every column of H^*_{sec} is nonzero; and, moreover, by the above lemma, every two columns of H^*_{sec} must be different, provided one is not the negative of another. Hence, for a given r, the maximum number of columns in the H^*_{sec} will be equal to $\frac{1}{2}(3^r - 1)$. Out of $\frac{1}{2}(3^r - 1)$ columns, r columns are used as a negative identity matrix. It can be seen that the maximum number of information bits to be encoded for a given r is equal to $\frac{1}{2}(3^r - 1) - r$. Moreover, if k is less than $\frac{1}{2}(3^r - 1) - r$, then k out of $\left(\frac{1}{2}(3^r - 1) - r\right)$ vectors will be selected in such a way that the vectors with more 0s will be selected first (Note that vectors in H^*_{sec} containing more 0s will result in a low complexity of encoding/decoding). A procedure to generate H^*_{sec} is given as follows.

PROCEDURE I.

Step 1: Let $k(r)$ denote the number of information (check) digits. For a given k, find the minimum r to satisfy:

$$\frac{1}{2}\left(3^r - 2r - 1\right) \geq k. \quad (1)$$

Step 2: For the r found in Step 1, find the minimum r', where $2 \leq r' \leq r$, to satisfy:

$$\sum_{i=2}^{r'} \binom{r}{i} 2^{i-1} \geq k.$$

Step 3: Let S_1 be the set containing all r-digit vectors which contain j_1 1s, j_2 – 1s, and $(r - j_1 + j_2)$ 0s, where $2 \leq j_1 + j_2 \leq r'$ and $0 \leq j_2 < \left\lceil \frac{j_1 + j_2}{2} \right\rceil$, and $\lceil x \rceil$ is the smallest integer greater than or equal to x.

Step 4: Let S_2 be the set containing all r-digit vectors which contain j_3 1s, j_3 – 1s and $(r - 2 \times j_3)$ 0s, where $2 \leq 2 \times j_3 \leq r'$. Partition S_2 into S_{21} and S_{22} such that

1) $S_2 = S_{21} \cup S_{22}$,
2) for every element $s \in S_{21}$, we can find an element $t \in S_{22}$ such that $s + t = 0$, where + is the vector addition.

Step 5: $S = S_{21} \cup S_1$.

Step 6: Let $|S|$ denote the cardinality of S. If $|S| - k > 0$, then delete $|S| - k$ elements from S arbitrarily.

Step 7: H^*_{sec} is an $r \times (k + r)$ dimensional matrix. In the transpose of H^*_{sec}, $\left[H^*_{sec}\right]^T$, each element in S is used as a row vector in the first k rows, the last r rows of $\left[H^*_{sec}\right]^T$ is the negative identity matrix.

We now use an example to illustrate the construction of an H^*_{sec}.

EXAMPLE 1. Let us construct an H^*_{sec}-matrix for $k = 26$ by using Procedure I. Since $k = 26$, the minimum r to satisfy the inequality:

$$\frac{1}{2} \times \left(3^r - 2r - 1\right) \geq 26$$

is equal to 4. In addition, we find that $r' = 3$.

The elements in S_1 are listed in the following table.

j_1	j_2	Elements in S_1
2	0	1 1 0 0, 1 0 1 0, 1 0 0 1,
2	0	0 1 1 0, 0 1 0 1, 0 0 1 1,
3	0	0 1 1 1, 1 0 1 1, 1 1 0 1, 1 1 1 0,
2	1	0 1 1 –1, 1 0 1 –1, 1 1 0 –1, 1 1 –1 0,
2	1	0 1 –1 1, 1 0 –1 1, 1 –1 0 1, 1 –1 1 0
2	1	0 –1 1 1, –1 0 1 1, –1 1 0 1, –1 1 1 0

The elements in S_2 are shown as follows.

Elements in S_{21}	Elements in S_{22}
–1 1 0 0, –1 0 1 0, –1 0 0 1	1 –1 0 0, 1 0 –1 0, 1 0 0 –1
0 –1 1 0, 0 –1 0 1, 0 0 –1 1	0 1 –1 0, 0 1 0 –1, 0 0 1 –1

An H^*_{sec} is constructed as follows:

$H^*_{sec} =$

$$\begin{bmatrix}
1&1&1&0&0&0&0&1&1&1&0&1&1&1&0&1&1&1&0&-1&-1&-1&-1&-1&0&0&-1&0&0&0\\
1&0&0&1&1&0&1&0&1&1&1&0&1&1&1&0&-1&-1&-1&0&1&1&1&0&-1&0&0&-1&0&0\\
0&1&0&1&0&1&1&1&0&1&1&1&0&-1&-1&-1&0&1&1&1&0&1&0&1&1&-1&0&0&-1&0\\
0&0&1&0&1&1&1&1&1&0&-1&-1&-1&0&1&1&1&0&1&1&1&0&0&0&0&1&0&0&0&-1
\end{bmatrix}$$

By using LHC, r additional check digits will be used. The relationship between k and r and the ratio between k and r in LHC are shown in Table 2.

TABLE 2
THE RELATIONSHIP BETWEEN k AND r IN LHC

k	5 ~ 10	11 ~ 36	37 ~ 116	117 ~ 358
r	3	4	5	6
$\frac{r}{k}$	0.6 ~ 0.30	0.36 ~ 0.11	0.14 ~ 0.04	0.05 ~ 0.02

We now consider the error detecting/correcting capability of the LHC.

THEOREM 1. *The code constructed by using the H^*_{sec} has the minimum Hamming distance* 3.

PROOF. We need to prove that every linear combination of two columns of H^*_{sec} is linearly independent. Let c_i and c_j be two distinct columns in H^*_{sec}. We need to show that

$$a_1 c_i = a_2 c_j \quad \text{iff} \quad a_1 = a_2 = 0,$$

where a_1 and a_2 are real numbers.

If $a_1 = a_2 = 0$, then $a_1 c_i = a_2 c_j$. We now show that $a_1 c_i = a_2 c_j$ implies that $a_1 = a_2 = 0$. We prove this by contradiction. Assuming that there exists $a_1 \neq 0$ or $a_2 \neq 0$ to satisfy $a_1 c_i = a_2 c_j$. By Lemma 1, we have $a_1 = a_2$, or $a_1 = -a_2$. Two cases are considered.

1) $a_1 = a_2$, i.e., $c_i = c_j$. In Step 3, Step 4, and Step 7 of the Procedure I, vectors in S_1, in S_{21}, and in the identity matrix are distinct. Hence, $c_i \neq c_j$ is valid for all $i \neq j$.
2) $a_1 = -a_2$, i.e., $c_i = -c_j$. In Step 3, vectors in S_1 have the property that the number of positive 1s is greater than that of –1s. In Step 4, vectors with the same number of 1s and –1s are selected into S_2. S_2 is partitioned into S_{21} and S_{22} in such a way that if c_i is in S_{21}, then $-c_i$ is in S_{22}. In Step 7, vectors in the identity matrix are used. Therefore, we can see that if c_i is selected, then $-c_i$ is not selected.

Since we cannot find $a_1 \neq 0$ or $a_2 \neq 0$ to satisfy $a_1 c_i = a_2 c_j$, we conclude that $a_1 c_i = a_2 c_j$ implies that $a_1 = a_2 = 0$. □

3.2 A Single Error Correcting/Double Error Detecting (SEC/DED) Code

In this subsection, we construct a code with the minimum distance 4, which can correct all single errors and detect all double errors simultaneously. Similar to the *lengthened Hamming code*, the parity-check matrix of this code contains 0, 1, and –1 so that encoding/decoding requires only additions/subtractions. We now introduce Procedure II to construct an $H^*_{sec/ded}$. Since the minimum Hamming distance is equal to 4, a linear combination of any two vectors in $H^*_{sec/ded}$ cannot be the third one. To construct such an $H^*_{sec/ded}$, Procedure II selects all r-bit vectors with the following properties: 1) the number of nonzero entries must be odd and is not equal to 3; 2) each vector contains exactly a single –1.

PROCEDURE II.

Step 1: For a given k, find the smallest r to satisfy the following inequality:

$$\sum_{i \geq 5,\, i=\text{odd}}^{r} i \times \binom{r}{i} \geq k. \tag{2}$$

Step 2: Let S be the set containing all r-digit vectors which have odd weight except three, provided that each vector contains exactly one –1.

Step 3: Let $|S|$ denote the cardinality of S. If $|S| - k > 0$, then delete $|S| - k$ elements from S arbitrarily.

Step 4: $H^*_{sec/ded}$ is an $r \times (k + r)$ dimension matrix. In the transpose of $H^*_{sec/ded}$, $\left[H^*_{sec/ded}\right]^T$, each element in S is

used as a row vector in the first k rows, the last r rows of $\left[H^*_{sec/ded}\right]^T$ is the negative identity matrix.

We now use an example to illustrate the construction of an $H^*_{sec/ded}$.

EXAMPLE 2. Let us construct an $H^*_{sec/ded}$-matrix for $k = 15$ by using Procedure II. For $k = 15$, the smallest r to satisfy the following inequality:

$$\sum_{i\geq 5,\, 1=\text{odd}}^{r} i \times \binom{r}{i} \geq 15$$

is 6. An $H^*_{sec/ded}$ is shown as follows:

$H^*_{sec/ded} =$

$$\begin{bmatrix}
-1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}$$

THEOREM 2. *The code constructed by using the $H^*_{sec/ded}$ has the minimum Hamming distance* 4.

PROOF. We need to prove that the code constructed by using the $H^*_{sec/ded}$ has the minimum distance 4. To do so, we prove a more general result that any three distinct columns of $H^*_{sec/ded}$ are linearly independent over the real number field. Let c_i, c_j, and c_k, be three distinct columns in $H^*_{sec/ded}$, and let $W(c_i)$, ($W(c_j)$ and $W(c_k)$) be the number of nonzero digits in the vector c_i (c_j and c_k). We need to show that

$$a_1c_i + a_2c_j = a_3c_k \quad \text{iff} \quad a_1 = a_2 = a_3 = 0,$$

where a_1, a_2, and a_3 are real numbers.

If $a_1 = a_2 = a_3 = 0$, then $a_1c_i + a_2c_j = a_3c_k$. We now show that $a_1c_i + a_2c_j = a_3c_k$ implies that $a_1 = a_2 = a_3 = 0$. The following cases are considered:

Case 1: $a_1 = 0$, or $a_2 = 0$, or $a_3 = 0$. If $a_1 = 0$, then $a_2c_j = a_3c_k$ implies $a_2 = a_3 = 0$ since c_j and c_k are nonzero and distinct, and $c_j \neq -c_k$. Similarly, if $a_2 = 0$, $a_1c_i = a_3c_k$ implies $a_1 = a_3 = 0$; and if $a_3 = 0$, $a_1c_i = a_2c_k$ implies $a_1 = a_2 = 0$.

Case 2: $a_1 \neq 0$ and $a_2 \neq 0$ and $a_3 \neq 0$. In this case, we prove that $a_1c_i + a_2c_j \neq a_3c_k$. For the sake of simplicity, we prove that

$$b_1c_i + b_2c_j \neq c_k$$

where

$$b_1 = \frac{a_1}{a_3} \text{ and } b_2 = \frac{a_2}{a_3}.$$

Two subcases are considered:

Case A: $W(c_i + c_j) = W(c_i) + W(c_j)$. In this subcase, since $W(c_i)$ and $W(c_j)$ are odd, $W(b_1c_i + b_2c_j)$ is even. However, $W(c_k)$ is odd, and so $b_1c_i + b_2c_j \neq c_k$.

Case B: $W(c_i + c_j) < W(c_i) + W(c_j)$. Since the nonzero entries of c_k contain a single -1 and even number of 1s, $b_1c_i + b_2c_j = c_k$ only if $W(b_1c_i + b_2c_j)$ is odd, and $b_1c_i + b_2c_j$ contains exactly one -1. Let the bit position of the -1 entry of c_i denote by l_i.

case a: $b_1 = b_2$.

1) **The l_ith bit of $c_j = -1$.** Hence, the nonzero entries of $b_1c_i + b_2c_j$ contain $-2b_1$ and b_1, whereas the nonzero entries of c_k contain only -1 and 1. Thus, $b_1c_i + b_2c_j \neq c_k$.
2) **The l_ith bit of $c_j = 0$.** If $W(b_1c_i + b_2c_j)$ is odd, then the nonzero entries of $b_1c_i + b_2c_j$ contain $-b_1$ and $2b_1$. Since the nonzero entries of c_k contain only -1 and 1, $b_1c_i + b_2c_j \neq c_k$.
3) **The l_ith bit of $c_j = 1$.** Since $W(c_i) \neq 3$ and $W(c_j) \neq 3$, then $W(b_1c_i + b_2c_j) \neq 1$. In addition, if $W(b_1c_i + b_2c_j)$ is odd, then the nonzero entries of $b_1c_i + b_2c_j$ have the same sign, or contain b_1 and $2b_1$, or contain $-b_1$ and $2b_1$. Therefore, $b_1c_i + b_2c_j \neq c_k$.

case b: $b_1 = -b_2$.

In this case, we only consider $b_1 > 0$ and similar arguments are valid for $b_1 < 0$. Three possibilities are considered.

1) **The l_ith bit of $c_j = 1$.** Hence, if $W(b_1c_i + b_2c_j)$ is odd, then the nonzero entries of $b_1c_i + b_2c_j$ contain $-2b_1$ and b_1, whereas the nonzero entries of c_k contain -1 and 1. Thus, $b_1c_i + b_2c_j \neq c_k$.
2) **The l_ith bit of $c_j = 0$.** If $W(b_1c_i + b_2c_j)$ is odd, then $b_1c_i + b_2c_j$ contains more than one $-b_1$ or contains $-b_1$ and $2b_1$. Since the nonzero entries of c_k contain only a single -1 and 1s, $b_1c_i + b_2c_j \neq c_k$.
3) **The l_ith bit of $c_j = -1$.** Since c_k contains only a single negative entry (-1), if $b_1c_i + b_2c_j$ contains only a single negative entry, then $W(b_1c_i + b_2c_j)$ is even. Since $W(c_k)$ is odd, and hence $b_1c_i + b_2c_j \neq c_k$.

Similar arguments are valid for $b_1 < 0$.

case c: $b_1 \neq b_2$ and $b_1 \neq -b_2$.

In this subcase, since $b_1 \neq b_2$ and $b_1 \neq -b_2$, only one of them can be either 1 or -1. Hence, if $b_1c_i + b_2c_j = c_k$, then there does not exist an m and an n such that $c_{im} = 0$, $c_{jm}=1$, $c_{in} = 1$, and $c_{jn} = 0$, where c_{im} and c_{in}(c_{jm} and c_{jn}) are respectively the mth and nth bit of c_i(c_j). Three possibilities are considered.

1) **$W(c_i) = W(c_j)$.** Clearly, $W(c_i) \geq 5$ and $W(c_j) \geq 5$ here. All nonzero entries of c_i and c_j must be overlapped. Then, $b_1 + b_2 = 1$ and $-b_1 + b_2 = -1$, or $b_1 + b_2 = 1$ and $b_1 - b_2 = -1$. Thus, $b_2 = 0$ or $b_1 = 0$. This contradicts to the assumption that $b_1 \neq 0$ and $b_2 \neq 0$. Thus, $b_1c_i + b_2c_j \neq c_k$.
2) **$W(c_i) > W(c_j)$.** Since $W(c_i) > W(c_j)$, then

$W(c_i) \geq W(c_j) + 2$. This implies $b_1 = 1$ or $b_1 = -1$. In order to have $b_1c_i + b_2c_j = c_k$, if $b_1 = 1$, then at least one of the following equations hold: $-b_1 - b_2 = -1$, $b_1 + b_2 = 1$, and $b_1 - b_2 = 1$; whereas if $b_1 = -1$, then $-b_1 - b_2 = -1$. However, all of them imply $b_2 = 0$. Contradiction. Thus, $b_1c_i + b_2c_j \neq c_k$.

3) $W(c_i) < W(c_j)$. The argument is similar to 2). □

The relationship between k and r and the ratio between k and r in this new code are shown in Table 3.

TABLE 3
THE RELATIONSHIP BETWEEN k AND r IN THE SEC/DEC CODE

k	6 ~ 30	31 ~ 112	113 ~ 436	437 ~ 787
r	6	7	8	9
$\frac{r}{k}$	1 ~ 0.20	0.26 ~ 0.06	0.07 ~ 0.02	0.02 ~ 0.01

3.3 Fault-Tolerant Matrix Operations

By using the proposed encoding/decoding schemes, we design new ABFT techniques for computing fault-tolerant matrix operations. For example, we encode matrix-vector multiplication as described in the following equation.

$$C^* = \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_{k-1} \\ c_k \\ cr_1 \\ \cdots \\ cr_r \end{bmatrix} = A^* \times B = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ & & \cdots & \\ a_{k-1,1} & a_{k-1,2} & \cdots & a_{k-1,m} \\ a_{k,1} & a_{k,2} & \cdots & a_{k,m} \\ ar_{1,1} & ar_{1,2} & \cdots & ar_{1,m} \\ & & \cdots & \\ ar_{r,1} & ar_{r,2} & \cdots & ar_{r,m} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_{m-1} \\ b_m \end{bmatrix} \quad (3)$$

where

$$ar_{i,j} = \sum_{n=1}^{k} h_{i,n} \times a_{n,j} \qquad 1 \leq j \leq m,\ 1 \leq i \leq r.$$

From the above discussion we can see that the check digits are formed by adding/subtracting a subset of elements to be encoded. Thus, we call these schemes as **partial check-sum**.

4 Overhead Evaluation

In this section, we examine overheads of ABFT techniques introduced by different encoding/decoding schemes. For the sake of simplicity, we consider a matrix-vector multiplication, and similar results can be obtained for other matrix operations. Note that when an ABFT technique is incorporated, the overhead includes encoding, decoding, and matrix-vector multiplication due to check digits. We consider the overhead in terms of the additional operations and the hardware overhead when the proposed algorithms are mapped into a fault-tolerant processor array.

4.1 Single Error Correction

In this subsection, we compare the overhead introduced by the LHC with that by the single error correcting code constructed by *weighted check-sum* schemes. The overhead of existing encoding/decoding schemes [8], [13], [10] in terms of additional operations is shown in Table 1. Since the complexity of a multiplication is the same as that of a division, the overhead in [12] is considered as the overhead of existing *weighted check-sum* schemes as shown in Table 1. When the LHC is used, r check digits, $ar_{j,i}$ $(1 \leq i \leq n, 1 \leq j \leq r)$, are used for each column as shown in (3). Since the complexity of an addition is the same as that of a subtraction, a subtraction is considered as an addition in the following. To construct $ar_{j,i}$ and cr_i, $c \times k$ additions are required, where $c \times k$ is the average number of nonzero entries in a row of H_{sec}, and the average c is equal to 0.6. In addition, the check digits in A^* will introduce additional $r \times k$ multiplications and $r \times (k-1)$ additions. The overhead is also shown in Table 4.

Theoretically, the complexity of a multiplication is an order of magnitude higher than that of an addition. Hence, from Table 4, the overhead of an ABFT constructed by the LHC has the complexity of $O(k \times r) = O(k \log k)$, compared to the existing schemes with complexity of $O(k^2)$. In practice, only a finite word length is used, e.g., 32-bit. If l is the word length, then a multiplication is at least l times more complex than an addition. Therefore, we consider the complexity of overhead in terms of additions, and define the overhead ratio (OR) as follows:

$$OR(d_{min}) = \frac{Overhead_{new}(d_{min})}{Overhead_{wcs}(d_{min})}$$

where d_{min} is the minimum distance of a code. When $d_{min} = 3$, we have

TABLE 4
OVERHEAD COMPARISONS WHEN $d_{min} = 3$ (SINGLE ERROR CORRECTION) WHERE r SATISFIES (1) AND r IS IN AN ORDER OF $\log_3 k$ (1)

	encoding	matrix-vector multiplication	decoding	total
weighted check-sum	k^2 multiplications $2k(k-1)$ additions	$2k$ multiplications $2(k-1)$ additions	k multiplications $2(k-1)$ additions	$k^2 + 3k$ multiplications $2k^2 + 2k$ additions
LHC	$c \times r \times k^2$ additions	$r \times k$ multiplications $r \times (k-1)$ additions	$c \times r \times k$ additions	$r \times k$ multiplications $r \times (c \times k^2 + (1+c) \times k - 1)$ additions

TABLE 5
OVERHEAD COMPARISONS WHEN $d_{min} = 4$ (SEC)/DED CODE) WHERE r SATISFIED (2)

	encoding	matrix-vector multiplication	decoding	total
weighted check-sum	$2k^2$ multiplications $3k(k-1)$ additions	$3k$ multiplications $3(k-1)$ additions	$2k$ multiplications $3(k-1)$ additions	$2k^2 + 5k$ multiplications $3k^2 + 3k$ additions
the SEC/DED code	$r \times k^2$ additions	$r \times k$ multiplications $r \times (k-1)$ additions	$r \times k$ additions	$r \times k$ multiplications $r \times (k^2 + 3k - 1)$ additions

$$OR(3) = \frac{Overhead_{sec}}{Overhead_{wcs}(3)} = \frac{r \times (c \times k^2 + k \times l + (1+c) \times k - 1)}{(k^2 + 3k) \times l + 2k^2 + 2k}. \quad (4)$$

For a 32-bit and a floating-point system, l is respectively equal to 32 and 24. The comparisons in terms of overhead for a 32-bit and for a floating-point system are shown in Fig. 2.

4.2 Single Error Correction/Double Error Detection

In this subsection, we compare the overhead introduced by the SEC/DED code with that by the single error correcting and all double detecting code constructed by *weighted check-sum* schemes. When $d_{min} = 4$, the overhead of two different approaches is shown in Table 5.

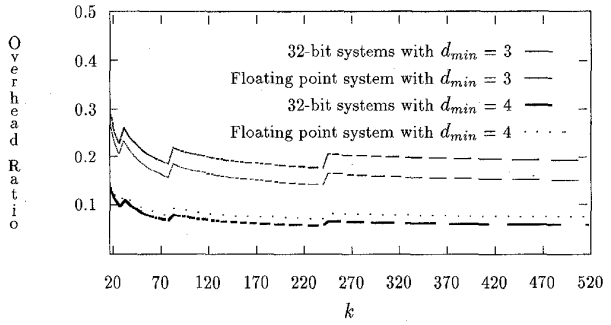


Fig. 2. Overheard ratio in terms of additional operations.

When $d_{min} = 4$, we have

$$OR(4) = \frac{Overhead_{sec/ded}}{Overhead_{wcs}(4)} \leq \frac{r \times (k^2 + k \times l + 3k - 1)}{(2k^2 + 5k) \times l + 3k^2 + 3k} \quad (5)$$

For a 32-bit and a floating-point system, the overhead ratios are also shown in Fig. 2.

We now consider the hardware overhead of the proposed schemes and existing techniques. Note that an important design rule for mapping an encoding/decoding algorithm into an array processor is that extra delays cannot incur a throughput degradation. In [19], existing and proposed encoding/decoding techniques are respectively mapped into an array processor. The hardware redundancy ratio, *HRR*, which is the ratio of the additional hardware required to implement fault tolerance and the logic required to carry out the matrix operations without fault tolerance. The *HRR*s for a 24-bit and a 32-bit system using *LHC* are shown in Fig. 3, which is significantly less than that of existing techniques. The detailed discussion can be found in [19].

5 CONCLUDING REMARKS

In this paper, we have proposed two new encoding/decoding schemes which can be used to design fault-tolerant matrix operations for array processors and multiprocessor systems. These two schemes enable a processor array either to tolerate all single faults or to tolerate all single faults and simultaneously detect all double faults. The major advantage of using these codes is the simplicity of encoding/decoding. Since only additions/subtractions are used in encoding/decoding, the total number of additional operations due to the incorporation of fault tolerance can be reduced to less than 30% of that used in existing schemes. Moreover, for matrix operations, it has been shown in [1], [14] that generalized linear codes should be used in ABFT techniques. In this paper, we introduce error detecting/correcting codes in $GF(3)$ with the minimum Hamming distance 3 (4) to construct efficient ABFT to tolerate all single faults (to tolerate all single faults and simultaneously detect all double faults). Therefore, we expect that error detecting/correcting codes in $GF(3)$ with the minimum distance d_{min}, where $d_{min} \geq 5$, could be used to construct efficient ABFT techniques which can tolerate two or more faults simultaneously.

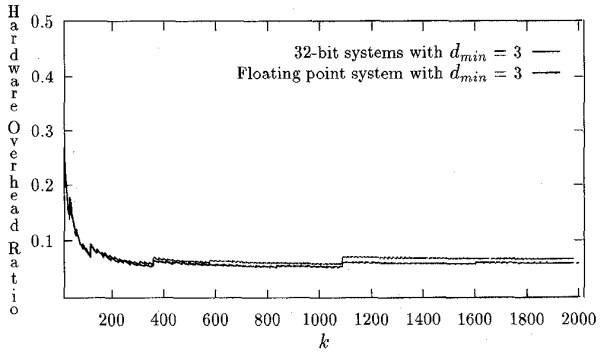


Fig. 3. Hardware overhead ratio.

ACKNOWLEDGMENTS

We are grateful to the valuable comments from the anonymous referees that significantly improved the quality of this paper.

REFERENCES

[1] C.J. Anfinson and F.T. Luk, "A Linear Algebraic Model of Algorithm-Based Fault Tolerance," *IEEE Trans. Computers*, vol. 37, no. 12, pp. 1,599-1,604, Dec. 1988.

[2] P. Banerjee and J.A. Abraham, "Bounds on Algorithm-Based Fault Tolerance in Multiple Processor Systems," *IEEE Trans. Computers*, vol. 35, no. 6, pp. 296-306, June 1986.

[3] P. Banerjee et al., "An Evaluation of System-Level Fault Tolerance on Intel Hypercube Multiprocessor," *Proc. FTCS-18*, pp. 362-367, 1988.

[4] D.L. Boley and F.T. Luk, "A Well Conditioned Checksum for Algorithmic Fault Tolerance," *Integration, the VLSI J.*, Elsevier Science Publishers, vol. 12, pp. 21-32, 1991.

[5] C.Y. Chen and J.A. Abraham, "Fault Tolerance Systems for the Computation of Eigenvalues and Singular Values," *Proc. SPIE, Advanced Algorithms and Architectures for Signal Processing*, vol. 696, pp. 222-227, 1986.

[6] D. Gu, D.J. Rosenkrantz, and S.S. Ravi, "Design and Analysis of Test Schemes for Algorithm-Based Fault Tolerance," *Proc. FTCS-20*, pp. 106-113, 1990.

[7] K.H. Huang and J.A. Abraham, "Algorithm-Based Fault-Tolerance for Matrix Operations," *IEEE Trans. Computers*, vol. 33, no. 6, pp. 518-528, June 1984.

[8] J.Y. Jou and J.A. Abraham, "Fault Tolerant Matrix Arithmetic and Signal Processing on Highly Concurrent Computing Structures," *Proc. IEEE*, vol. 74, pp. 732-741, May 1986.

[9] J.Y. Jou and J.A. Abraham, "Fault Tolerant FFT Networks," *IEEE Trans. Computers*, vol. 37, no. 5, pp. 548-561, May 1988.

[10] K.Y. Lin, H. Krishna, and J.B. Wang, "Algebraic Techniques for Algorithm-Based Fault Tolerance in Signal Processing Systems," *Proc. 23rd Asilomar Conf. Signals, Systems, and Computers*, Oct. 1989.

[11] F.T. Luk, "Algorithm-Based Fault Tolerance for Parallel Matrix Equation Solvers," *Proc. SPIE Real Time Signal Processing*, vol. 564, pp. 49-53, 1985.

[12] F.T. Luk and H. Park, "An Analysis of Algorithm-Based Fault Tolerance Techniques," *J. Parallel and Distributed Computing*, vol. 5, pp. 172-184, 1988.

[13] F.T. Luk and H. Park, "A Fault Tolerance Matrix Triangularizations on Systolic Arrays," *IEEE Trans. Computers*, vol. 37, no. 11, pp. 1,434-1,438, Nov. 1988.

[14] V.S.S. Nair and J.A. Abraham, "General Linear Codes for Fault-Tolerant Matrix Operations on Processor Arrays," *Proc. FTCS-18*, pp. 180-185, June 1988.

[15] V.S.S. Nair and J.A. Abraham, "Hierarchical Design and Analysis of Fault-Tolerant Multiprocessor Systems Using Concurrent Error Detection," *Proc. FTCS-20*, pp. 130-137, June 1990.

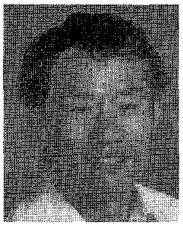
[16] T.R.N. Rao, *Error Coding for Arithmetic Processors*. New York: Academic Press, 1974.

[17] A.L.N. Reddy and P. Banerjee, "Algorithm-Based Fault Detection for Signal Processing Applications," *IEEE Trans. Computers*, vol. 39, no. 11, pp. 1,304-1,308, Nov. 1990.

[18] D.L. Tao, C.R.P. Hartmann, and Y.S. Chen "A Novel Concurrent Error Detection Scheme for FFT Networks," *Proc. FTCS-20*, pp. 114-121, June 1990.

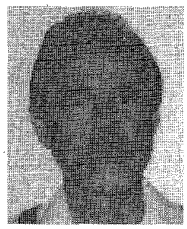
[19] D.L. Tao and C.R.P. Hartmann, "Algorithm-Based Fault Tolerance for Matrix Operations," CEAS Technical Report 581, Dept. of Electrical Engineering, State University of New York at Stony Brook, Apr. 1990.

[20] B. Vinnakota and N.K. Jha, "A Dependence Graph-Based Approach to the Design of Test Schemes for Algorithm-Based Fault Tolerant Systems," *Proc. FTCS-20*, pp. 122-129, 1990.



D.L. Tao received the BS degree in electrical engineering from the Beijing Institute of Posts and Telecommunications in 1982, the MS in electrical engineering from Syracuse University in 1984, and the PhD in computer engineering from Syracuse University in 1988.

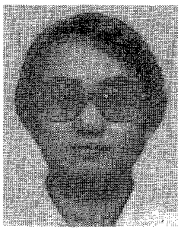
Dr. Tao is now with the Department of Electrical Engineering at the State University of New York at Stony Brook. His research interests include fault-tolerant computing, parallel processing, and multimedia technology. He is also working and consulting for Canon U.S.A., Northrop-Grumman, and Symbol Technologies.



C.R.P. Hartmann received the BS and MS degrees in electrical engineering from Instituto Technologico de Aeronautic (ITA), Sao Jose dos campos (S.P.), Brazil, in 1963 and 1966, respectively, and the PhD degree from the University of Illinois, Urbana-Champaign, in 1970.

Dr. Hartmann was an instructor at ITA in 1964 and 1965 and a research assistant at the Coordinated Science Laboratory at the University of Illinois from 1966 to 1970. He is currently director and a professor at the School of Computer and Information Science at Syracuse University in New York. He is interested in error control for digital systems, fault detection in digital circuits, fault-tolerant computer design, analysis and design of algorithms, and data compression.

Dr. Hartmann is a member of Phi Kappa Phi and Brazil's Conselho Regional de Engineharia de Arquitecura. From 1981 to 1983, he was the associate editor for coding theory for *IEEE Transactions on Information Theory*.



Yunghsing S. (Sam) Han received the BS and MS degrees in electrical engineering from the National Tsing Hua University in Hsinchu, Taiwan, Republic of China, in 1984 and 1986, respectively, and the PhD degree from the School of Computer and Information Science at Syracuse University in New York in 1993.

Dr. Han was a lecturer at the Ming-Hsin Engineering College, Hsinchu, Taiwan, from 1986 to 1988; and a teaching assistant and research assistant in the School of Computer and Information Science, Syracuse University, from 1989 to 1992 and 1992 to 1993, respectively. He is now an associate professor in the Department of Electronic Engineering at the Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. His research interests are in error-control coding and fault-tolerant computing.