CHOOSING SMALL WEIGHTS FOR MULTIPLE ERROR DETECTION

RICHARD P. BRENT

Computer Sciences Laboratory, Australian National University
Canberra, ACT 2601, Australia

FRANKLIN T. LUK

School of Electrical Engineering, Cornell University
Ithaca, New York 14853

CYNTHIA J. ANFINSON

Center for Applied Mathematics, Cornell University
Ithaca, New York 14853

ABSTRACT

The weighted checksum technique has been demonstrated to be effective in multiple error detection. It has been shown that, in order to guarantee error detection, the chosen weight vectors must satisfy some very specific properties about linear independence. Previous weight generating methods that fulfill the independence criteria have problems with numerical overflow. We present a new scheme that generates weight vectors to meet the requirements about independence and to avoid the difficulties with overflow.

1. INTRODUCTION

The importance of solving signal processing problems in real time and the development of VLSI and wafer-scale technology have led to research in systolic arrays and algorithms. There is a need for high-performance digital signal processing systems which are extremely reliable. Algorithm-based fault tolerance has been proposed to meet this reliability need, since the most common alternative of duplicating hardware is often too expensive to be practical.

The weighted checksum scheme, originally developed by Huang and Abraham [4] and Jou and Abraham [5], has been demonstrated to be effective as a means for low-cost error protection. It has been shown that for an appropriate set of weights in a distance $d+1$ code, one may detect a maximum of d errors [2]. By an appropriate set of weights we mean a set of weight vectors which satisfy certain conditions about linear independence. Previously, appropriate sets of weight vectors have been proposed which are powers of integers [1], [5], [7]; these suffer from the fact that the weights can become very large. We propose here a technique for generating an appropriate set of weights that are reasonably sized.

This paper is organized as follows. We first review the weighted checksum scheme and discuss an important theorem that guarantees multiple error detection. The results of this theorem place restrictions on the weight vector selection. Then we present a new procedure for generating weight vectors to satisfy these conditions. Finally, we discuss the problem of numerical overflow and demonstrate how our method overcomes this difficulty.

2. DESCRIPTION OF WEIGHTED CHECKSUM PROCEDURE

For error protection in matrix computing, Huang and Abraham [4] and Jou and Abraham [5] developed the checksum and weighted checksum schemes. Their techniques have been shown to be very amenable to systolic computing. A linear algebraic interpretation of algorithm-based fault tolerance has been proposed in [4] and [5], and in the work of Anfinson and Luk [2]. The model unifies algorithm-based fault tolerance with conventional coding theory, hence allowing parallels to be drawn between the results. Results in coding theory rely upon the use of finite fields, and the linear algebraic model extends the theory to n dimensional vector spaces over infinite fields. We now present a synopsis of these results.

2.1 Review of the checksum method

Jou and Abraham [5] introduced weighted checksums to handle the case of multiple error detection and correction. If d suitably chosen weighted checksum rows (or columns) are appended to the matrix A, then we may detect up to d errors [2]. The checksum scheme of Huang and Abraham [4] is the weighted checksum scheme with $d = 1$. For unique $n\times1$ weight vectors $w^{(i)}$, $i = 1, \cdots, d$, with elements $w_j^{(i)}$, $j=1, \cdots, n$, we define the $(n+d)\times n$ weighted column checksum matrix A_{cw} as

$$A_{cw}=\begin{bmatrix} A \\ -w^{(1)T}A \\ -w^{(2)T}A \\ \cdot \\ \cdot \\ \cdot \\ -w^{(d)T}A \end{bmatrix}.$$

The $n\times(n+d)$ weighted row checksum matrix may be defined in a similar manner [5].

2.2 Linear algebraic model

The weighted checksum matrix H is similar to the parity-check matrix from coding theory [6]; vectors in the null space of H are called code vectors and they make up the code C. The matrix H is made up of the d weight vectors $w^{(i)}$, $i = 1, \cdots, d$, with elements $w_j^{(i)}$ for $j = 1, \cdots, n$ and a $d\times d$ identity matrix.

Definition 1. The $d\times(n+d)$ weighted checksum matrix H is given by

$$H=\begin{bmatrix} w_1^{(1)} & w_2^{(1)} & \cdots & w_n^{(1)} & 1 & 0 & \cdots & 0 \\ w_1^{(2)} & w_2^{(2)} & \cdots & w_n^{(2)} & 0 & 1 & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & 0 & 0 & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot & \cdot & \cdots & \cdot \\ w_1^{(d)} & w_2^{(d)} & \cdots & w_n^{(d)} & 0 & 0 & \cdots & 1 \end{bmatrix}. \qquad (2.2.1)$$

Definition 2. The code space C of H consists of all vectors which are elements of the null space $N(H)$ of H, where $N(H) = \{ x : Hx = 0 \}$.

We define a metric on the domain of H to derive a notion of distance on the code space C. It is easily checked that the metric satisfies the properties of a distance [2].

Definition 3. The distance between two vectors v and w in the domain of H, $dist(v, w)$, equals the number of components in which v and w differ.

Definition 4. The distance of the code space C is the minimum of the distances between all possible pairs of nonzero vectors in $N(H)$; i.e., distance of $C = \min \{dist(v,w) : v, w \text{ in } N(H), v \neq 0, w \neq 0\}$.

Definition 5. Let x be in $N(H)$, and $\hat{x}$ be a possibly erroneous version of x. Define the syndrome vector s by $s = H\hat{x}$.

We will use the following definition of detection throughout this work. Let α denote the total number of errors that have occurred. We will say that a coding scheme can *detect* α errors if the syndrome vector s is nonzero whenever $1 \leq \alpha \leq d$. The next result gives us the conditions under which d errors may be detected in a distance $d+1$ code.

Theorem 1. If every set of d columns of H is linearly independent, then the distance of the code is $d+1$ and a maximum of d errors can be detected.

Proof. See reference [2]. □

3. WEIGHT SELECTION PROCEDURE

We now present our technique for generating weight vectors which satisfy the conditions of Theorem 1, and avoid the overflow problem from which previous schemes have suffered. Our scheme is dependent on the values of n and d, but as the step of generating the weight vectors is pre-processing, it will not slow down the execution of the algorithm. Our method uses modular arithmetic, but not in the same way as suggested by Abraham et al. [4], [5].

Consider the values of n and d to be known and fixed. The size of the input matrix determines n, and d is selected by the user according to a model which determines the expected number of errors. Choose a prime number p such that $p > n+d$. It is well known that for any prime p there exists a finite field of p elements [3]. Let α be a primitive element of the finite field of p elements. Recall that a primitive element β of a finite field with q elements is an element with order $q-1$, i.e., the smallest integer m for which $\beta^m = 1 \ mod \ q$ is $m = q-1$. It is easy to see that powers of β generate all the nonzero elements of the field.

Let B be the following matrix

$$B = (b_{ij}) = (\alpha^{(i-1)(j-1)}), \text{ for } 1 \leq i \leq d \text{ and } 1 \leq j \leq n+d. \tag{3.1}$$

Partition B as

$$B = \left[V \ W \right]. \tag{3.2}$$

Here V has dimension $d \times n$ and W has dimension $d \times d$. Now, in order to get the desired form of H, we multiply the matrix $B \ mod \ p$ by the matrix $W^{-1} \ mod \ p$:

$$H = W^{-1} \left[V \ W \right] mod \ p = \left[W^{-1}V \ mod \ p \mid I \right]. \tag{3.3}$$

The matrix H now has the form of equation (2.2.1). To make the above procedure valid, we must prove the following. First, we need to show that W is nonsingular so that the multiplication in (3.3) is valid. Then we need to prove

that every set of d columns of the resulting matrix H is linearly independent, so that we can guarantee the detection of a maximum of d errors.

Proposition 1. The matrix $B \ mod \ p$, where B is given by equation (3.2), satisfies the condition that every set of d columns is linearly independent.

Proof. Let B_d denote a $d \times d$ matrix whose columns are any d columns chosen from B, i.e.,

$$B_d = \left[b_{j_1} \ b_{j_2} \ . \ . \ . \ b_{j_d} \right], \text{ where } 1 \le j_1 < j_2 < \cdots < j_d \le d+n.$$

Hence, each column b_{j_i} has the form

$$b_{j_i} = \begin{bmatrix} \beta_i^0 \\ \beta_i^1 \\ . \\ . \\ . \\ \beta_i^{d-1} \end{bmatrix}, \text{ where } \beta_i = \alpha^{j_i - 1} \text{ for } i = 1, 2, \cdots, d.$$

Thus, B_d is a Vandermonde matrix. If $\det(B_d \ mod \ p) = 0$ then

$$\prod_{k>l} (\beta_k - \beta_l) = 0 \ mod \ p,$$

which is equivalent to

$$\beta_k - \beta_l = 0 \ mod \ p, \text{ for some } k > l,$$

i.e., $\alpha^{j_k - j_l} = 1 \ mod \ p$. It follows that

$$j_k \ge p - 1 + j_l \ge p > n + d,$$

a contradiction. Therefore, $B_d \ mod \ p$ has linearly independent columns. □

Corollary 1. The $d \times d$ matrix $W \ mod \ p$, where W is defined in equation (3.2), is nonsingular.

Proof. Since any d columns of the matrix $B \ mod \ p$ are linearly independent, it follows by definition of $W \ mod \ p$ that it is nonsingular. □

Corollary 2. The matrix H, given by equation (3.3), satisfies the hypotheses of Theorem 1.

Proof. From Corollary 1 we know that the multiplication by the matrix $W^{-1} \ mod \ p$ is valid. Since H is obtained from $B \ mod \ p$ by premultiplication by this nonsingular matrix, the result follows. □

We note that the choice of W is by no means unique. We could choose any d columns and permute the matrix into the form given by equation (2.2.1); for our sole objective is to satisfy the hypotheses of Theorem 1. We now illustrate some of the above ideas in the following example.

Example 1. Suppose we are given a problem where $n = 8$ and d is selected to be 4. Thus, by fulfilling the requirements of Theorem 1 we will be able to detect a maximum of 4 errors in each column of the input matrix. Since $n + d = 12$, we take $p = 13$, and $\alpha = 2$ is a primitive root of p. From equation (3.2) we see that the matrix $B \ mod \ p$ has the form

$$B \ mod \ p = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 3 & 6 & 12 & 11 & 9 & 5 & 10 & 7 \\ 1 & 4 & 3 & 12 & 9 & 10 & 1 & 4 & 3 & 12 & 9 & 10 \\ 1 & 8 & 12 & 5 & 1 & 8 & 12 & 5 & 1 & 8 & 12 & 5 \end{bmatrix}. \quad \square$$

As the parity-check matrix H must take the form of equation (2.2.1), we need to find the inverse of $W \ mod \ p$. If we use the Gauss-Jordan method for finding matrix inverses, we can do the inversion $mod \ p$ since the procedure consists of only elementary row operations. We then perform the multiplication $W^{-1} \ V \ mod \ p$. Alternatively, we can find the LU factors of W via Gaussian elimination $mod \ p$, which also involves only elementary row operations, and then compute each column of the product $W^{-1} \ V \ mod \ p$ using the LU factors. While computing $W^{-1}V \ mod \ p$ is expensive, we once again point out that it is all pre-processing; it will not add extra running time to the algorithm. Furthermore, all the elements of the parity-check matrix H are bounded above by p.

Example 2. Using the same matrix B as generated in Example 1, we see that $W \ mod \ p$ is the matrix

$$W \ mod \ p = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 9 & 5 & 10 & 7 \\ 3 & 12 & 9 & 10 \\ 1 & 8 & 12 & 5 \end{bmatrix}.$$

Hence, we can find the inverse of $W \ mod \ p$

$$W^{-1} \ mod \ p = \begin{bmatrix} 8 & 5 & 6 & 8 \\ 6 & 11 & 0 & 12 \\ 5 & 0 & 9 & 7 \\ 8 & 10 & 11 & 12 \end{bmatrix}.$$

Multiplying out $W^{-1} \ V \ mod \ p$, we get the product

$$W^{-1} \ V \ mod \ p = \begin{bmatrix} 1 & 2 & 12 & 4 & 7 & 6 & 1 & 10 \\ 3 & 7 & 12 & 11 & 12 & 12 & 9 & 5 \\ 8 & 6 & 12 & 5 & 2 & 8 & 7 & 11 \\ 2 & 12 & 4 & 7 & 6 & 1 & 10 & 1 \end{bmatrix}. \quad \square$$

4. COMPARISON WITH PREVIOUS WORK

As mentioned previously, in [4] and [5], methods were proposed using modular arithmetic to compute weighted checksums. Let us examine how our method differs from these previous schemes. Note that for means of comparison we will consider fixed point arithmetic since it is used in [4] and [5]. Let r denote the word length. Huang and Abraham [4] suggested that we compute the actual checksum values modulo 2^r. In Jou and Abraham [5], a similar technique is proposed where the checksum column is computed modulo 2^r and the weighted checksum column is computed modulo N, where N is the largest prime less than 2^{r+1}. They prove that for this choice of weights, the scheme detects errors as desired. One difference between the methods is that our new one is all pre-processing while theirs add extra time to the algorithm as we must compute the checksums using modular arithmetic. Furthermore, the prime that is selected by Jou and Abraham is somewhat larger than the prime we come up with. For example, in [5], if the word length $r = 32$ then the prime $N = 8589934583$. Our scheme is relatively independent of the word length; it depends upon the values of n and d. For $n = 1000$, a large value in light of the dimensions required by current signal processing problems, and $d = 50$, the prime $p = 1051$, which is much smaller than the Jou-Abraham value of N.

We next compare the sizes of the elements in the weighted checksum matrix H for various weight generating schemes. Suppose that $n = 500$ and $d = 10$. Then the range of values for i and j is $i = 1, \cdots, 10$ and $j = 1, \cdots, 500$. Jou and Abraham proposed the set of weights $w_j^{(i)} = 2^{(i-1)(j-1)}$ [5]. While easily implemented as shifts, this set of weights become extremely large very quickly, resulting in overflows. For $i = 10$ and $j = 500$, we get $w_j^{(i)} = 2^{4491}$. Another suggestion was to let $w_j^{(i)} = j^{i-1}$ [7]. It is also seen that for reasonable sized problems these weights will get large very rapidly, although not as quickly as the previous technique. Using the same example, with $i = 10$ and $j = 500$, we have $w_j^{(i)} = 500^9$. For our new scheme, the smallest prime p satisfying $p > n+d = 510$ is $p = 521$. Therefore, every element $w_j^{(i)}$ will be bounded above by 521. Thus, we see that for this moderately sized problem, our new scheme generates a parity-check matrix whose elements are likewise of moderate size. We have avoided the overflow problems of the previously proposed schemes.

5. CONCLUSIONS

We have presented a new method for generating the weighted checksum matrix H, in particular a scheme that generates a matrix whose elements are bounded in size by the smallest prime p such that $p > n+d$. Since we usually consider $d \ll n$, p is approximately $O(n)$. To generate the matrix H involves only pre-processing steps, and thus will take no additional time in the algorithm.

6. ACKNOWLEDGEMENTS

The work of F.T. Luk was supported in part by the SDIO/IST under an Army Research Office contract DAAL03-86-G-0016, and that of C.J. Anfinson by an Army Science and Technology Graduate Fellowship.

7. REFERENCES

[1] C.J. Anfinson, R.P. Brent and F.T. Luk, "A theoretical foundation for the weighted checksum scheme," *Proc. SPIE, Vol. 975, Advanced Algorithms and Architectures III*, Aug. 1988, Paper 2.

[2] C.J. Anfinson and F.T. Luk, "A linear algebraic model of algorithm-based fault tolerance," *IEEE Trans. Comput., Vol. C-37, No. 12*, pp. 1599-1604, Dec. 1988.

[3] I.N. Herstein, *Topics in Algebra, Second Edition*, John Wiley and Sons, New York, 1975.

[4] K.H. Huang and J.A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Trans. Comput., Vol. C-33, No. 6*, pp. 518-528, June 1984.

[5] J.Y. Jou and J.A. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures, *Proc. IEEE, Vol. 74, no. 5, Special Issue on Fault Tolerance in VLSI*, pp. 732-741, May 1986.

[6] S. Lin and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.

[7] F.T. Luk, "Algorithm-based fault tolerance for parallel matrix equation solvers," *Proc. SPIE, Vol. 564, Real Time Signal Processing VIII*, pp. 49-53, Aug. 1985.