Comments

A Note on "A Systematic (12,8) Code for Correcting Single Errors and Detecting Adjacent Errors"

Mario Blaum, Jehoshua Bruck, and Ludo Tolhuizen

Abstract—**Schwartz and Wolf give a parity check matrix for a systematic (12,8) binary code that corrects all single errors and detects eight of the nine double adjacent errors within any of the three 4-bit nibbles. Here, we present a parity check matrix for a systematic (12,8) binary code that corrects all single errors and detects *any* pair of errors within a nibble.**

Index Terms—**Adjacent errors, error-correcting/detecting codes, 4-bit nibble, parity-check matrix, systematic codes.**

In [3], the authors studied the following problem: Given a byte stored as three 4-bit nibbles, such that 8 of the bits carry information and 4 are redundant, encode the information bits in such a way that any single error will be corrected and any two adjacent bits in error in a nibble will be detected. The authors come up with a solution that in fact corrects any single error and detects 8 out of the 9 cases of double adjacent errors in a nibble.

Note that the code given in [3] is not optimal-it can detect only 8 out of the 9 patterns of double adjacent errors within a nibble. Moreover, the authors in [3] claim that it is not possible to find a systematic code that can detect the 9 patterns of double adjacent errors within a nibble. This claim, however, is true for the particular case in which the first 8 bits carry the information and the last 4 bits carry the redundancy. Any code equivalent to this one (i.e., its coordinates are a permutation of the coordinates of the original code) is also systematic [2].

In this note, we present a counterexample to the claim in [3], namely, a systematic (12,8) code that can correct any single error and detect any of the 9 patterns of double adjacent errors within a nibble. Moreover, our code detects *any* pattern of double errors (adjacent or not) within a nibble.

As pointed out above, an (n,k) code is systematic if there are k positions in which the codewords carry the information, and the remaining $n-k$ positions are redundant [2]. Consider a (12,8) code with the following parity check matrix:

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

Since the columns are all distinct, this code is single-error correcting [2]. As we can see, the 4×4 submatrix of H at locations 2, 3, 6, and 10 is the identity matrix. Therefore, these coordinates correspond to the redundancy and the information is carried in bits 1, 4, 5, 7, 8, 9, 11, and 12.

Notice that the syndromes corresponding to double errors within nibbles (i.e., the EXCLUSIVE-OR's of all possible pairs of columns of H within a nibble) are 0111, 1100, or 1011. These vectors do not correspond to any of the columns of H, therefore, they cannot be confused with a syndrome corresponding to a single error. Hence, the code can correct any single error and detect any double error within a nibble, providing a counterexample to [3]. Let us point out that our results have been generalized in a recent paper [1].

Manuscript received May 7, 1991.

M. Blaum and J. Bruck are with IBM Research Division, Almaden Research Center, San Jose, CA 95120.

L. Tolhuizen is with Philips Research Laboratories, 5600 JA Eindhoven, The Netherlands.

IEEE Log Number 9209032.

References

[1] T. Etzion, "Optimal codes for correcting single errors and detecting adjacent errors," *IEEE Trans Inform. Theory,* vol. 38, no. 4, pp. 1357–1360, July 1992.

[2] J. H. van Lint, *Introduction to Coding Theory.* New York: Springer-Verlag, 1982.

[3] J. W. Schwartz and J. K. Wolf, "A systematic (12,8) code for correcting single errors and detecting adjacent errors," *IEEE Trans. Comput.,* vol. 39, no. 11, pp. 1403–1404, Nov. 1990.

Comments on "Synthetic Traces for Trace-Driven Simulation of Cache Memories"[1]

Syed Masud Mahmud

Abstract—**A number of errors have been discovered in the above paper[1] The authors of paper[1] have corrected some of these errors and presented in correspondence.[2] The remaining errors are corrected and presented in this correspondence.**

Index Terms— **Cache memories, performance analysis, trace-driven simulation, LRU stack model, program locality.**

I. Introduction

Several incorrect equations and expressions have been found in the above paper[1]. Some of these incorrect equations are corrected and presented in correspondence. There are still two errors which have not been corrected yet. These remaining errors are indicated and corrected in this comment.

Error 1: On p. 393 of the above paper[1], a synthetic trace generation algorithm is shown in pseudo-Pascal for the infinite memory model. There is a mismatch between the number of open and close parentheses in the following assignment statement of the algorithm.

$$\text{index} := \text{round}((U/(A^{\theta})/\theta))^{(1/(1-\theta}))$$

[1] D. Thiebaut, J. L. Wolf, and H. S. Stone, *IEEE Trans. Comput.*, vol. 41, pp. 388–410, Apr. 1992.

[2] D. Thiebant, J. L. Wolf, and H. S. Stone, "Corrigendum to 'Synthetic trace for trace-driven simulation of cache memories,'" *IEEE Trans. Comput.*, vol. 42, pp. 635–636, May 1993.

Manuscript received April 13, 1993; revised August 6, 1993.

The author is with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202.

IEEE Log Number 9212757.