A Novel Approach to Improving Burst Errors Correction Capability of Hamming Code

Jianwu Zhao
School of Automation Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan Province, P.R. China

Yibing Shi
School of Automation Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan Province, P.R. China

***Abstract*—Error detection and correction is critical to accurate data transmission, storage and retrieval. Error Correction Coding (ECC) has been a crucial part of data transmission or storage. In high-reliability applications, the single error correction and double error detections (SEC-DED) Hamming code may not provide adequate protection against burst errors. This makes multiple-error correction (MEC) highly desirable. This paper proposed a novel approach to improving burst errors correction capability of the extended Hamming Code, with the target objective of minimizing the redundancy, while retaining code rate as maximum as possible. Design algorithms of Hamming encoding and decoding are proposed in this paper, and the simplicity and ease of their implementations are demonstrated with example.**

I. INTRODUCTION

There are various types of interferences in transmission channel, which can negatively impact data transmission between the transmitting station and the receiving station, and both random and burst errors have occurred during transmission. In long term storage of data, the contents of storage device can be corrupted for various reasons [1]. In order to detect or correct transmission or storage errors, some additional, redundant bits, called "check bits", can be added to a number of data bits to construct a codeword.

Our concern in this paper is exclusively binary forward error correcting (FEC) block codes in data transmission. This means that the symbol alphabet consists of just two symbols (which we denote 0 and 1), that the receiving station can correct a transmission error without asking the transmitting station for more information or for a retransmission, and that the transmissions consist of a sequence of fixed length blocks. More specifically, Forward error correction, the ability of the receiving station to correct a transmission error, can increase the throughput of a data link operating in a noisy environment. The transmitting station must append information to the data in the form of error correction bits, but the increase in frame length may be modest relative to the cost of retransmission. An example of an FEC code is known as the Hamming code. For example, the (7, 4, 3) Hamming code is able to protect a four bit information (data bits) from a single error in a codeword by adding three redundant bits to the data bits.

Hamming Code is still widely used in computing, telecommunication, and other applications including data compression, popular puzzles, and turbo codes. There is, however, a disadvantage associated with Hamming code. While a single bit error is correctable, errors typically occur in bursts that may only be dealt with by Hamming code with large Hamming distances. What is required by the industry is a coding process that will correct error bursts, and need less redundant information to be added. The paper proposed a novel approach to correcting burst errors in the extended Hamming code, with the target objective of minimizing the redundancy, while retaining code rate as maximum as possible. Furthermore, the paper gives the design algorithms of Hamming encoding and decoding which are easy to be implemented with less time consuming in programming.

The paper is organized into six sections. An effective construction method for the (16, 11, 4) extended Hamming code is given in Section II. Section III presents the approach to interlacing and rebuilding the Hamming code for correcting burst errors, and error mechanisms are discussed, while Section IV provides the details of the design algorithms. Section V gives example, experimental results and their discussions based the (16, 11, 4) extended Hamming code. The concluding remarks are presented in Section VI.

II. CONSTRUCTION OF THE EXTENDED HAMMING CODE

The following notation is used throughout this paper. Let k represent the number of information or data bits. Let r represent the number of additional parity check bits. The result of appending the computed parity check bits to the data bits is called Hamming codeword. The size of the codeword is obviously $k+r$, denoted by n and a Hamming codeword is described by the ordered set (n, k, r). Because the r parity check bits must check themselves as well as the data bits, the value of parity check bit vector, interpreted as

The research is based upon work supported by Program for New Century Excellent Talents in University (NCET-05-0804).

an integer, must range from 0 to $k+r$, which is $k+r+1$ distinct values [2]. Because r bits can distinguish 2^r cases, we must have

$$2^r \geq k + r + 1 . \quad (1)$$

This is known as the Hamming rule [3].

The key to constructing Hamming coding is the use of extra parity check bits to allow the identification of a single error. To construct a Hamming code, we have to calculate the r parity check bit vector from k bit data. There are different known approaches to the construction of Hamming code. We proposed an effective method for constructing the extended identity check matrix of the extended Hamming code, and then we can work out check codes from this matrix. The following is to illustrate how to construct the identity check matrix of the (16, 11, 4) extended Hamming code.

The (16, 11, 4) extended Hamming code consist of 11 information bits, 4 parity-check bits and 1 overall parity bit. The size of (16, 11, 4) code word is obviously 15. The values, range from 1 to size of code word, are denoted as binary numbers that range from 0000 to 1111. We can construct a matrix, denoted by H, whose columns are the binary numbers increasing from left to right. Original form of matrix H:

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (2)$$

The identity check matrix of the (16, 11, 4) extended Hamming code can be constructed from this matrix H: add a zero column on the left, then a row of all 1's on the top. The positions of data bits and parity check bits can be reorganized according to the programming requirement. Finally we get the identity-check matrix of the (16, 11, 4) extended Hamming code, as follows:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

A (16, 11, 4) Hamming codeword would be constructed by inserting parity check bits and an overall parity check bit in bit positions 1, 2, 4, 8, and 16, with the data bits occupying bit positions 3, 5, 6, 7, 9, …, 15. The check codes are respectively C0=0xAB61, C1=0xCDA2, C2=0xF1C4, C3=0xFE08, and C4=0xFFFF. The algorithm is presented below.

A. Algorithm for construcing the identity check matrix

For any integer p there exists a (n, k, r) extended Hamming code, where:

$$n = 2^p - 1; k = 2^p - 1 - p \quad (4)$$

The detailed formulation of the various steps involved in the implementation is given below.

Step 1) Create a matrix with dimension $p \times (2^p - 1)$, which each column is to be different, i.e. all binary combinations should be used, except all-zero's one.

Step 2) Reorder the columns in order to achieve an identity matrix in the right hand side.

Step 3) Take first $(2^p - 1) - p$ columns of the matrix (all columns except those creating the identity part), and then add a zero column (overall parity-check column) to this matrix on the left.

Step 4) Append this matrix with an overall parity-check column to an identity matrix with dimensions $p \times p$.

Step 5) Add a row of all 1's to the matrix with dimensions $(p+1) \times 2^p$ on the top.

Step 6) The check code is equal to the value of each row of the extended identity check matrix.

If the bits of a code word are numbered from 1 to n, parity check bits are located at all power-of-two bit positions, with the data bits interspersed in the remaining bit positions.

III. Implementation of Design Approach to Improve Burst Errors Correction Capability

For data transmission on serial channels, the most prevalent errors are multi-bit bursts. These multi-bit errors make parity worthless, and severely limit the effectiveness of single bit correcting Hamming codes [4]. The proposed approach to correct burst error is the interleaving reorganization of Hamming codewords, with the target objective of minimizing the redundancy, while retaining code rate as maximum as possible. More specifically, the error control approach is that at most 1 bit is affected in each code word, while a burst error of length n occurs in the Hamming code.

From the practical standpoint of data transmission, a (7, 4, 3) Hamming code is not a good choice, because it involves non-standard character length. Designing a suitable code requires that the ratio of parity check bits to data bits and the processing time involved to encode and decode the data stream be minimized, a coding method that efficiently handles 8 bit data items is desirable. Furthermore, the Hamming rule shows that four parity bits can provide error correction for five to eleven data bits, with the latter being a perfect code and analysis shows that redundant bits introduced to the data stream is modest for the range of data bits available (11 bits 36% overhead, 8 bits 50% overhead, 5 bits 80% overhead).

We illuminate the approach by using an example of the (16, 11, 4) extended Hamming code. A byte access fetches one bit from each of the (16, 11, 4) extended Hamming

codeword. Thus, a byte access references 8 bits of codeword that are physically somewhat separated. With bits interleaved in that way, if a few close-together bits in the same byte are corrupted during data transmission under interferences, a few (16, 11, 4) codeword will have single-bit error, which can be corrected. These advantages, however, come at the cost of failing to correct multiple single-bit errors in separate byte, which result in consecutive bits error in a code word. Namely, this technique degrades the ability to correct single bit error (random errors) in Hamming codewords. On the other hand, the frame format for the asynchronous data transmission is like this: one start bit, eight data bits and one stop bit. Both start bit and stop bit are produced by a universal asynchronous receiver transmitter (UART). So both start bit and stop bit aren't encoded in order to identify a frame data. If the start bit or stop bit is corrupted by interferences in transmission channel, the corrupted data will not also be restored using the interleaved Hamming code.

When we construct useful Hamming code, we should decide whether or not the interleaving technique is used based on error mechanisms. More specifically, Error mechanisms are often not completely random, and may affect a number of adjacent bits giving a burst error. In contrast, random errors can be distributed throughout the length of the code word. Gaussian noise, as in the space channel, tends to cause random errors; bust noise, as in cable or radio systems, will produce error bursts [5] [6].

IV. Hamming Encoding and Decoding Algorithms for Correcting Burst Errors

This section presents the algorithm for the (16, 11, 4) extended Hamming encoding and decoding processes with the afore-said the interleaving reorganization technique. Furthermore, the (16, 11, 4) extended Hamming code is of practical use because the code rate defined as the number of information bits divided by the code length is 0.7.

A. Algorithm for the encoding

The length of original data block processed at a time is 11 bytes. The length of the extended Hamming code to encode these 11 bytes original data is 16 bytes. The algorithm for the encoding is presented below.

Step 1) Take out the 11 bytes data from the original data and then get 8 data with 11 bit width by simply merging.

Step 2) Compute the values of even check bits and overall even check bit from the check codes (C0, C1, C2, C3, and C4) determined by the extended identity-check matrix, and then respectively put these even check bits in position 1, 2, 4, 8, and 16. All other bit positions are for the data to be encoded.

Step 3) Repeat Step 2. Process other 11 bits data until all data are encoded.

Step 4) Reorganize the Hamming codewords by the use of interleaving technique that a byte access fetches one bit from each of the extended Hamming code codeword.

B. Algorithm for the decoding

The algorithm for decoding is presented below.

Step 1) Restore the encoded data block by the use of de-interleaving technique.

Step 2) Fetch a code word into buffer and reset the error flag bit.

Step 3) Extract the overall parity bit from a code word.

Step 4) Compute the value of the check code (C0, C1, C2, C3, and C4) of received code word.

Step 5) If the value of C4, C3, C2, and C1 are equal to zero, then go to Step 8.

Step 6) If the value of the overall parity bit is equal to zero, then set up an error flag.

Step 7) Locate and fix a bad bit from the value of the check code (C0, C1, C2, C3, and C4) of received code word. A bad bit is located by using lookup table method.

Step 8) Output 8 bits data and the error flag.

Step 9) Repeat Step 2 - 8. Process other code words until all codewords are decoded.

V. Example and Experimental Results

In this section, we gave an example and experimental results of the Hamming encoding and decoding testing program in assembly language and assume the use of 8-bit microcomputer AT89C55. This Hamming encoding and decoding program can also directly be embedded in user's program and the length of original data block and memory access address can be assigned by user.

The length of original testing data is 11 bytes and the length of Hamming codewords is 16 bytes. The pertinent memory assignments for the extended Hamming encoding and decoding testing program are listed in the following Table I.

TABLE I. Memory Assignments for Testing Program

Data Class	Physical Memory Area
Original Testing Data	0x2000 – 0x200A
Encoded Data	0x2020 – 0x202F
Interleaved Data	0x2040 – 0x204F
Decoded Data	0x2060 – 0x206A

The testing program automatically generates 11 bytes original testing data that range from 0 to 10. These testing data are encoded into the 16 bytes (16, 11, 4) extended Hamming codewords. First, we show that the burst errors correction capability of the extended Hamming code is

severely limited without using the interleaving reorganization technique. Note that the above encoding program encodes 11 bits of an original data into 16 bits of a codeword and then outputs 8 bits of data. Multiple errors are inserted to simulate an error burst occurring. The consecutive 2 bits errors are inserted to change the content value of codeword transmitted from 0x01 to 0x31 which results in an error occurring of the decoded data. Fig. 1 shows the simulation result.

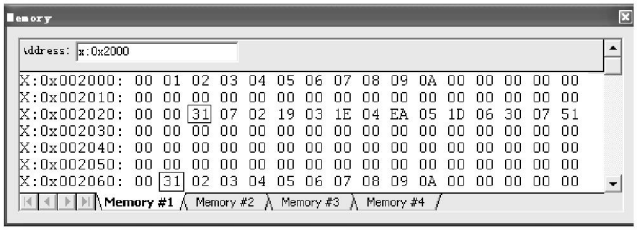

Figure 1. The contents of memory without using interleaving technique

These codewords are dealt with the afore-said the interleaving reorganization technique before they are transmitted. During the encoding, the contents of memory are shown in Fig. 2.

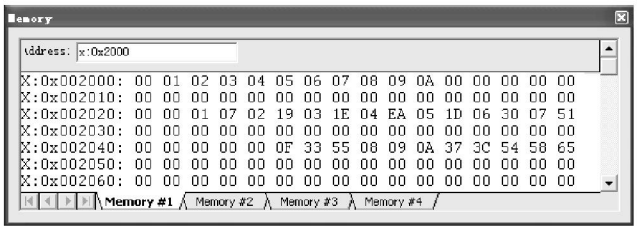

Figure 2. Encoded data in memory

For example, suppose the above data 0x0F is sent and burst errors occur such that the data 0xF0 is received. There are the consecutive 8-bits errors. The contents of memory are shown in Fig. 3.

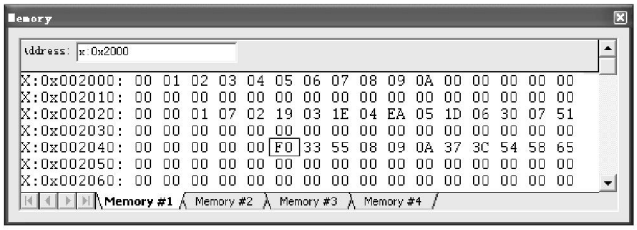

Figure 3. Corrupted encoded data in memory

Finally, the corrupted data are decoded, and we can find that the above multiple errors can be corrected. The contents of memory are shown in Fig. 4.

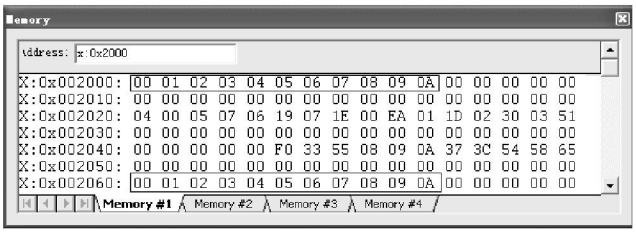

Figure 4. Decoded data in memory

The above analysis illustrate that the approach proposed in the paper corrects the consecutive 8-bits errors which is the worst case in 8-bits data transmission. What is made clear by the example and experimental results is that research on multiple error correction technique has led to more efficient encoding scheme, thus codes that need minimum redundant information to be added for the better burst errors correction capability.

VI. CONCLUSION

In the paper, we analyze the advantage and disadvantage of the Hamming Codes and error mechanisms causing random errors and burst errors. We give an effective method to construct the extended identity check matrix of the extended Hamming code. Once the extended identity check matrix has been constructed, the corresponding check codes for implementing error detecting and correcting can be worked out. Because burst errors can be effectively randomized by the use of interleaving, burst errors correction capability of Hamming codes is greatly improved. The experimental results show that the efficiency of the approach proposed in the paper.

REFERENCES

[1] Etzion, Tuvi " Constructions for perfect 2-burst-correcting codes," IEEE Transactions on Information Theory, Vo6. 47, pp.2553-2555, 2001.

[2] Lin, Shu and Costello, Daniel J., Error Control Coding: Fundamentals and Applications. Prentice-Hall, 1983.

[3] MacWilliams, Florence J. and Sloane, Neil. J. A. The Theory of Error-Correcting Codes Part II. North-Holland, 1977.

[4] C. L. Chen and M. Y. Hsiao, "Error-Correcting Codes for Semiconductor memory applications: A State-of-the-Art review," IBM J. Res. Develop., Vol. 28, pp. 124-134, July 1984.

[5] D. Rossi, V. Dijk, R. Kleihorst, A. K. Nieuwland, and C. Metra, "Power Conssumption of Fault Tolerant Codes: the Active Elements," Proc, of Intentional On-Line Testing Symposium, pp.61-67, 2003.

[6] W. Gao and S. Simmons, "A study on the VLSI implementation of ECC for embedded DRAM," Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conf., Vol. 1, pp. 203-206, May 2003.