Fast Parallel Decoding of Double-Error-Correcting Binary BCH Codes

T. A. GULLIVER*
Department of Electrical and Electronic Engineering
University of Canterbury, Christchurch, New Zealand

W. LIN
Department of Systems and Computer Engineering
Carleton University, Ottawa, Ontario, Canada K1S 5B6

F. DEHNE
Department of Computer Science
Carleton University, Ottawa, Ontario, Canada K1S 5B6

(Received January 1997; accepted January 1998)

Abstract—This paper presents a new high speed parallel decoding algorithm for double-error-correcting binary BCH codes. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—BCH codes, Parallel decoding.

1. INTRODUCTION

The Bose-Chaudhuri-Hocquenghem (BCH) codes were independently discovered by Hocquenghem [1] and Bose and Ray-Chaudhuri [2] in 1959 and 1960, respectively. In the almost 40 years since then, they have become the most extensively studied class of error-correcting codes. The main reason for this popularity is the well-known algebraic decoding algorithms, chief of which is the Berlekamp-Massey algorithm. This allows decoding up to a designed distance, which makes them attractive for system developers. In addition, the designed distance of many BCH codes is optimal in the sense that it is the best possible minimum distance for any linear code with the same parameters. In this paper, we consider only binary BCH codes, since they are employed in the majority of applications. The codewords are assumed to be in systematic form, so that the information bits occupy the $k$ most significant positions.

Familiarity with basic coding principles is assumed. These can be found in virtually any text on coding theory, such as [3].

2. DECODING BCH CODES

Let $i(x)$ be the information polynomial, $c(x)$ the transmitted codeword polynomial, and $r(x)$ the received polynomial. In almost all cases, the first step in decoding is to compute the syndromes, $S_i$, from $r(x)$. This is achieved by evaluating $S_i = r(\alpha^i)$, where $\alpha$ is a root of the

*Formerly with the Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada K1S 5B6.

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada.
generator polynomial, \( g(x) \). For a \( t \) error-correcting code, \( t \) syndromes must be found. These syndromes are used to determine the error locations, which are denoted by a 1 in the error polynomial \( e(x) \). Decoding is then completed by adding \( e(x) \) and \( r(x) \) to obtain the codeword estimate \( \hat{c}(x) \). If decoding is successful (less than \( t \) errors have occurred), \( \hat{c}(x) = c(x) \).

If \( t = 1 \), \( S_1 \) can provide the error location directly. For multiple-error-correcting BCH codes, algebraic decoding consists of two additional steps.

1. Construct the error locator polynomial \( \sigma(x) \).
2. Find the error locations which are the roots of \( \sigma(x) \).

Since Step 2 requires a search of all possible error locations (Chien search), it becomes very slow for large \( n \).

**EXAMPLE 1.** Consider the two error-correcting \([15,7,5]\) BCH code with generator polynomial
\[
g(x) = m_1(x)m_3(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1.
\]

Let the received vector be
\[
r = 110111101011000.
\]

The syndromes are
\[
S_1 = r(\alpha) = \alpha^{11}, \quad S_3 = r(\alpha^3) = \alpha^5.
\]

The error locator polynomial is given by
\[
\sigma(x) = x^2 + S_1x + \frac{S_3}{S_1} + S_1^2
\]
\[
= x^2 + \alpha^{11}x + 1.
\]

A Chien (exhaustive) search is performed to find the roots of \( \sigma(x) \), which are \( \alpha^7 \) and \( \alpha^8 \). The error locations correspond to the powers of \( \alpha \), so the error vector is
\[
e = 000000011000000.
\]

This vector is added to the received vector to obtain the codeword estimate
\[
\hat{c} = r + e = 110111110011000.
\]

Fast decoding with a small number of parity symbols \((n - k)\) can be achieved via table lookup, whereby the syndromes are effectively an index into a table which holds the error vectors. For an \([n, k]\) BCH code, the size of this table is \(2^{n-k} \times k\) bits (if the code is in systematic form). For larger \( n \) and \( n - k \), the table size becomes unacceptable for practical applications. In these cases, an approach called step-by-step decoding can be employed [4]. This involves changing each bit in \( r(x) \) and determining whether the number of errors has decreased [4].

To correct all patterns of two or less errors in a double-error-correcting BCH code, the following relations between syndromes can be applied.

1. If \( S_1 = S_3 = 0 \), there are no errors.
2. If \( S_1 \neq 0 \) and \( S_3 = S_1^2 \), there is one error.
3. If \( S_1 \neq 0 \) and \( S_3 \neq S_1^2 \), there are two errors.
Fast Parallel Decoding

In addition, $S_1 = 0$ and $S_3 \neq 0$ indicates that three or more errors have occurred. Define the indicators $I_h$ as follows:

$$I_0 = 1 \iff S_1 = S_3 = 0,$$
$$I_1 = 1 \iff S_1 \neq 0, S_3 = S_1^3,$$
$$I_2 = 1 \iff S_1 \neq 0, S_3 \neq S_1^3,$$
$$I_3 = 1 \iff I_0 = I_1 = I_2 = 0.$$

Clearly the index, $h$, of the nonzero indicator denotes the number of errors in $r(x)$.

3. PARALLEL DECODING

The concept of step-by-step decoding is very simple: invert a position, recalculate the indicators, and compare them with the original values. If an erroneous bit in $r(x)$ has been inverted, the number of errors will be reduced by 1, as will the index of the nonzero indicator. If the indicator index stays the same or increases, the location is not in error.

Denote $S_h^i$ and $I_h^i$ as the syndromes and indicators when position $n - k + j$ of $r(x)$ is inverted (recall that the information bits occupy the $k$ most significant positions). The following procedure can be used to decode $r(x)$.

1. Calculate $S_1$ and $S_3$ and the indicators $I_0$, $I_1$, $I_2$, and $I_3$.
2. Calculate the syndromes $S_h^i$, $i = 1, 3$, $j = 0, 1, \ldots, k - 1$. 
3. Calculate the indicators $I_0^j$, $I_1^j$, $I_2^j$, and $I_3^j$.
4. Invert the bit in position $j \iff I_h = 1$ and $I_{h-1}^j = 1$ for some $1 \geq h \geq 2$.
5. Read out the decoded information bits from $\hat{c}(x)$.

If $I_0 = 1$, the information bits can be read out directly since $r(x)$ is a codeword. Also, if $I_3 = 1$, $r(x)$ contains more than two errors and cannot be decoded. In this case, a decoding failure should be declared, and if possible a retransmission requested. In all other cases, the above steps are executed.

A block diagram of the decoder is given in Figure 1. The received polynomial $r(x)$ enters at the left where it is stored in a buffer, and simultaneously the syndromes $S_1$ and $S_3$ are computed. These syndromes are then fed into the $k$ branches corresponding to each possible information bit (0 to $k - 1$). Branch $j$ corresponds to the inversion of bit $j$ so that $S_1^j = S_1 + \alpha^j$ and $S_3^j = S_3 + \alpha^{3j}$.

Thus, the new syndromes can be obtained simply by adding a constant to those previously obtained.

$I_0^j$ and $I_3^j$ are easily computed, so the most difficult portion of the decoding operation is determining if $S_3^j = (S_1^j)^3$. This can be done via a simple logic circuit which performs the cubing operation, and compares the result to $S_3^j$. The decision circuit then compares the indicators and outputs a 0 or 1 (corresponding to the bit in $\hat{c}(x)$) according to Step 4 above. This error vector is added to the received vector, and the corrected information bits, $\hat{c}(x)$, are output.

Note that if $I_0 = 1$ or $I_3 = 1$, no further decoding is required, so the buffer contents can be output immediately.

4. CONCLUSIONS

A new parallel decoder for double-error-correcting BCH codes has been presented. It is based on the step-by-step decoding algorithm. The required circuitry is quite simple and so is suitable for high speed applications. The extension to three or more error-correcting BCH codes is straightforward, but does involve an increase in decoder complexity.

REFERENCES