Parallel Error-Trapping and Error-Detection Decoding

Patrick Lee and Stanley Chang

Western Digital Corp., San Jose CA, 95119

Abstract
In this paper, we present a parallel decoding method for a concatenated outer (nearest the channel) error-correcting code (ECC) and inner (nearest the user) error-detecting code (EDC). This method can be used if (1) both the ECC and EDC are linear codes, (2) the ECC decoder can be implemented as an error-trapping decoder and (3) the EDC decoder is realized by a polynomial division circuit. Compared to a serial implementation, the parallel decoding has less decoding delay and requires very minimal additional hardware.

1. Introduction.

One method of improving the integrity of data transmitted over a noisy channel is to use a concatenated code consisting of an inner (nearest the channel) error-correcting code (ECC) and an outer (nearest the user) error-detecting (EDC) code. With this approach, the user data is first encoded by EDC encoder and then by the ECC encoder. As illustrated in Fig. 1, the data can be decoded serially. That is, the received data is first decoded by the ECC decoder, which hopefully corrects all the channel induced errors, and subsequently by the EDC decoder, which then checks for any undetected errors in the user data. In particular, note that the coding delay (i.e. time required by the decoder to process the data) for the serial approach is the sum of the delay for the ECC decoder and the EDC decoder.

Because of the structure of the ECC and EDC, the serial decoding approach is often the only means of recovering the data (for example, see [1]-[4]). However, as will be shown, if
(1) the ECC and EDC are both linear codes,
(2) the ECC decoder can be implemented as an error-trapping decoder, and
(3) EDC decoder can be implemented by polynomial division circuit defined over some finite field, GF(q), then decoding operations can be implemented using the parallel decoding structure shown in Fig. 2.

The parallel approach is attractive because it involves less coding delay than the serial approach with very minimal increase in additional hardware. Specifically, the worst case coding delay for the parallel approach is only slightly longer than the worst case coding delay for the ECC decoder. Further, it can be used with any ECC
and EDC which belong to the class of cyclic codes (including CRC's, BCH and Reed-Solomon codes) as well as some burst error-correcting codes.

In Section 2, we describe the parallel decoding algorithm when the received codeword is corrected by both a forward and a backwards error-trapping ECC decoder, and in Section 3, we present a mathematical justification for the algorithm.

2. The Parallel Decoding Algorithm.

The basic idea of the algorithm is as follows. First, we compute the syndrome for the EDC from the (uncorrected) received codeword, \( v(x) = c(x) + e(x) \), where \( c(x) \) is the transmitted codeword polynomial and \( e(x) \) is the error polynomial, using the usual division circuit. Specifically, we compute

\[
\sigma(x) = R_g(x)[e(x)] = R_g(x)[c(x)] + R_g(x)[e(x)] = R_g(x)[e(x)],
\]

where \( R_g(x)[e(x)] \) denotes the remainder when \( e(x) \) is divided by the EDC generator polynomial, \( g(x) = g_0 + g_1 x + \ldots + g_m x^m \).

If

\[
\sigma(x) = \sigma_0 + \sigma_1 x + \ldots + \sigma_m x^m \neq 0,
\]

then the received codeword must be corrected by the ECC decoder. When corrections are necessary, we can avoid having to recompute the EDC syndrome after the corrections by simply computing a new syndrome, \( \sigma(x) \), from the "correction" polynomial, \( e(x) \), generated by the ECC decoder. Explicitly, we compute

\[
\sigma(x) = R_g(x)[e(x)] \equiv \sigma_0 + \sigma_1 x + \ldots + \sigma_m x^m - 1,
\]

where

\[
e(x) = e_0 + e_1 x + \ldots + e_n x^n,
\]

and \( e_k \) is the error magnitude for the \( k^{th} \) symbol generated by the ECC decoder. Next, we compare \( s(x) \) and \( \sigma(x) \). If \( s(x) = \sigma(x) \), then we assume that all errors have been correctly decoded by the ECC decoder, otherwise we assume that the ECC decoder has miscarried the data. Further, since \( s(x) = \sigma(x) \) if and only if \( s(x) + \sigma(x) = 0 \) (modulo \( q \)), it follows that we can simplify the implementation by computing

\[
s(x) = R_g(x)[e(x) + s(x)].
\]

In this case, if \( s(x) = 0 \), we assume that the received data was correctly decoded, otherwise \( (s(x) \neq 0) \), we assume that the received data was miscarried. Thus, since this amounts to just adding the syndrome symbols (computed from the received codeword) to the appropriate correction symbols generated by the ECC decoder, the parallel decoding algorithm can be implemented as shown in Fig. 3.

![Figure 3. Parallel Decoding With Forward Error-Trapping ECC Decoder](image_url)

However, for shortened codes, it is sometimes more efficient to perform "backwards" error-trapping, in which the ECC decoder outputs the reciprocal correction polynomial,

\[
e^*(x) = x^m e(1/x) = e_0 + e_1 x + \ldots + e_n x^n.
\]

As a consequence, the ECC decoder must compute the syndrome given the sequence of ECC corrections in reverse order (i.e. instead of processing the sequence \( e_n, \ldots, e_0 \), the ECC decoder must process the sequence \( e_n, e_1, \ldots, e_0 \)). In this case, parallel decoding can be implemented as shown in Fig. 2. In particular, note for the backwards case, \( \sigma(x) \) is given by

\[
\sigma(x) = R_g(x)[x^{n-m-l}s(x) + e^*(x)].
\]
where

$$s^*(x) = x^{m-1}s(1/x) = s_{m-1} + s_{m-2}x + ... + s_0x^{m-1}$$

is the reciprocal polynomial of the (forward) syndrome, $s(x)$, and

$$g^*(x) = x^mg(1/x) = g_m + x^{m-1} + ... + g_0x^m$$

is the reciprocal of the generator polynomial, $g(x)$. That is, for backwards decoding, the syndrome bits are delivered in reverse order, where the first output bit of the EDC syndrome computer is $s_0$ (for forward decoding, the first output bit is $s_{m-1}$).

Thus, it follows immediately that parallel decoding is equivalent to serial decoding when the ECC decoder uses a forward error-trapping algorithm.

To show that the parallel algorithm also works when used in conjunction with a backwards ECC error-trapping decoder, it is sufficient to show that

$$R_{g(x)}[v(x) + e(x)] = R_{g(x)}[v(x)] + R_{g(x)}[e(x)]$$

if and only if

$$R_{g^*(x)}[x^{m-1}s^*(x) + e^*(x)] = 0.$$
However, since for any two polynomials, \( a(x) \) and \( b(x) \), of finite degree,

\[
[f(x)]^* = a(x) \quad \text{and} \quad [ab(x)]^* = a^*(x)b^*(x),
\]

where \( [f(x)]^* \) is the reciprocal polynomial of \( f(x) \), it follows that

\[
e(x) + e(x) = [e^*(x) + e^*(x)]^* = [q(x)g^*(x)]^* = q^*(x)g(x).
\]

Thus, \( e(x) + e(x) \) is divisible by \( g(x) \) and \( R_{g(x)}[e(x) + e(x)] = 0 \). Since using a similar argument shows the reverse is true, it then follows that parallel decoding in conjunction with a backwards error-trapping ECC decoder is equivalent to serial decoding.

References.


