A Simplified Procedure for Correcting Both Errors and Erasures of a Reed–Solomon Code Using the Euclidean Algorithm

T. K. Truong and I. S. Hsu
Communications Systems Research Section

W. L. Eastman
Mitre Corporation

I. S. Reed
University of Southern California

It is well known that the Euclidean algorithm or its equivalent, continued fractions, can be used to find the error locator polynomial and the error evaluator polynomial in Berlekamp's key equation needed to decode a Reed–Solomon (RS) code. In this article, a simplified procedure is developed and proved to correct erasures as well as errors by replacing the initial condition of the Euclidean algorithm by the erasure locator polynomial and the Forney syndrome polynomial. By this means, the errata locator polynomial and the errata evaluator polynomial can be obtained, simultaneously and simply, by the Euclidean algorithm only. With this improved technique the complexity of time-domain RS decoders for correcting both errors and erasures is reduced substantially from previous approaches. As a consequence, decoders for correcting both errors and erasures of RS codes can be made more modular, regular, simple, and naturally suitable for both VLSI and software implementation. An example illustrating this modified decoding procedure is given for a (15, 9) RS code.

I. Introduction

The Euclidean algorithm for solving the key equation for decoding both BCH and Goppa type codes was developed first by Sugiyama *et al.* [1]. Forney [2] defined an errata locator polynomial using what are now called Forney syndromes to correct both errors and erasures. Blahut [3, p. 258] showed that the errata locator polynomial can be computed directly by initializing Berlekamp's algorithm with the erasure locator polynomial.

Recently, Eastman[1] showed that the errata evaluator polynomial can be computed directly by initializing Berlekamp's

[1] W. L. Eastman, "Decoding Erasures," unpublished Mitre Corporation Report, Bedford, MA, 1986.

algorithm with the Forney syndrome polynomial. A proof of this new simplified decoding procedure is given in the present article. By this technique it is possible to compute the errata locator polynomial and the errata evaluator polynomial simultaneously from the Euclidean algorithm. It uses both the erasure locator polynomial and the Forney syndrome polynomial as initial conditions of the Euclidean algorithm.

This new Reed–Solomon (RS) decoder can be realized by a simplified pipeline architecture. An efficient time domain decoder can be developed for correcting both errors and erasures of RS codes. Such a decoding technique can be faster and simpler than previous methods [4].

II. The Time Domain Decoder for RS Codes

An algorithm is developed in [4] for time domain decoding RS codes to correct both errors and erasures by the use of continued fractions or its equivalent, the Euclidean algorithm. This algorithm is a modification of the Forney–Berlekamp method [2, 5]. The block diagram of such a decoding algorithm for a (255, 223) RS code over $GF(2^8)$ is depicted in Fig. 1. In this algorithm, the continued fraction algorithm is used to find the errata locator polynomial by replacing its initial condition by the erasure locator polynomial. The disadvantage of this algorithm is that after the errata locator polynomial $\tau(x)$ is obtained by continued fractions, a polynomial multiplication is still needed to compute the errata evaluator polynomial $A(x) = \lfloor S(x)\ \tau(x) \rfloor$ from the known errata locator polynomial and the syndrome polynomial $S(x)$, where $\lfloor x \rfloor$ denotes the principal part of x.

In this section, the above-mentioned algorithm is modified to correct both errors and erasures in the time domain decoding of RS codes by a new use of the Euclidean algorithm. In this new algorithm, depicted in Fig. 2, the Euclidean algorithm is used to solve the Berlekamp–Forney key equation for the errata locator polynomial and the errata evaluator polynomial directly and simultaneously. The advantage of this algorithm over previous methods [4] is that the separate computation of the errata evaluator polynomial, usually needed as in [4], can be avoided. This new decoding algorithm is highly suitable to both VLSI and software implementation.

First, let $GF(2^m)$ be a finite field of 2^m elements. Also, let $N = 2^m - 1$ be the length of the (N, I) RS code over $GF(2^m)$ with minimum distance d, where $I = N - (d - 1)$ denotes the number of m-bit message symbols and $d - 1$ denotes the number of parity symbols such that $d - 1$ is either an even or an odd integer.

Define the following five vectors:

$c = (c_0, c_1, \ldots, c_{N-1})$, code vector

$r = (r_0, r_1, \ldots, r_{N-1})$, received vector

$e = (e_0, e_1, \ldots, e_{N-1})$, error vector

$u = (u_0, u_1, \ldots, u_{N-1})$, erasure vector

$u = (\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{N-1})$, errata vector

These vectors are related by $\widetilde{u} = e + u$ and $r = c + u + e$.

Suppose that t errors and ν erasures occur in the received vector r and assume that $\nu + 2t \leqslant d - 1$. Next let α be a primitive element in $GF(2^m)$. Then $\gamma = \alpha^i$ is also a primitive element in $GF(2^m)$, where $(i, N) = 1$.

To minimize the complexity of an RS encoder, it is desirable that the generator polynomial be symmetric. If γ is a root of the code's generator polynomial, it is shown [6] that the generator polynomial $g(x)$ is symmetric, if and only if,

$$g(x) = \prod_{i=b}^{b+(d-2)} (x - \gamma^i) = \sum_{i=0}^{d-1} g_i x^i \tag{1}$$

where $g_0 = g_{d-1} = 1$ and b satisfies the equality $2b + d - 2 = 2^m - 1$.

The syndromes of the code are given by

$$S_{(b-1)+k} = \sum_{i=0}^{N-1} \tilde{u}_i \gamma^{i(b-1+k)} = \sum_{j=1}^{\nu+i} Y_j X_j^{(b-1)+k}$$

$$\text{for } 1 \leqslant k \leqslant d-1 \tag{2}$$

where X_j is either the j^{th} erasure or error location, and Y_j is either the j^{th} erasure or error magnitude. Define the sets, $\Lambda = \{X_i | X_i \text{ is an erasure location}\}$ and $\lambda = \{X_i | X_i \text{ is an error location}\}$. Also, it is not difficult to show, see [5], that

$$S(x) = \sum_{k=1}^{d-1} S_{(b-1)+k}\, x^{k-1}$$

$$= \sum_{j=1}^{\nu+t} \frac{Y_j X_j^b}{(1-X_j x)} - \sum_{j=1}^{\nu+t} \frac{Y_j\ X_j^{b+d-1} x^{d-1}}{(1-X_j x)} \tag{3}$$

Following [4] define four different polynomials in terms of sets Λ and λ as follows:

The erasure locator:

$$\Lambda(x) = \prod_{X_j \epsilon \Lambda} (1 - X_j x) = \prod_{j=1}^{\nu} (1 - X_j x)$$

$$= \sum_{j=0}^{\nu} (-1)^j \Lambda_j x^j \qquad (4a)$$

where $\Lambda_0 = 1$.

The error locator:

$$\lambda(x) = \prod_{X_j \epsilon \lambda} (1 - X_j x) = \prod_{j=1}^{t} (1 - X_j x)$$

$$= \sum_{j=0}^{t} (-1)^i \lambda_j x^j \qquad (4b)$$

where $\lambda_0 = 1$.

The errata locator:

$$\tau(x) = \Lambda(x)\,\lambda(x) = \prod_{j=1}^{\nu+t} (1 - X_j x)$$

$$= \sum_{j=0}^{\nu+t} (-1)^j \tau_j x^j \qquad (4c)$$

where $\tau_0 = 1$.

The errata evaluator:

$$A(x) = \sum_{j=1}^{\nu+t} Y_j X_j^b \Big(\prod_{i \neq j} (1 - X_i x)\Big) \qquad (4d)$$

In terms of the polynomials defined above, (3) becomes

$$S(x) = \frac{A(x)}{\tau(x)} + \frac{x^{d-1} \sum_{j=1}^{\nu+t} Y_j X_j^{b+d-1} \Big[\prod_{i \neq j} (1 - X_i x)\Big]}{\tau(x)} \qquad (5a)$$

or

$$S(x)\,\tau(x) = A(x) + x^{d-1} \sum_{j=1}^{\nu+t} Y_j X_j^{b+d-1} \Big[\prod_{i \neq j} (1 - X_i x)\Big] \qquad (5b)$$

From (5b) one obtains the congruence relation

$$S(x)\,\tau(x) \equiv A(x) \bmod x^{d-1} \qquad (6a)$$

Now define the set of formal power series

$$F = \left\{ \sum_{i=0}^{\infty} a_i x^{n-i} \mid a_i \in GF(2^m), \text{ and } n \text{ is an integer} \right\}$$

Since $\tau(x)$ in (6a) is a polynomial in x with the leading coefficient not equal to zero, it is not difficult to show that the inverse element $\tau^{-1}(x)$ always exists in the set of formal power series. Thus, (6a) can be solved for $S(x)$ to yield

$$S(x) \equiv \frac{A(x)}{\lambda(x)\,\Lambda(x)} \bmod x^{d-1} \qquad (6b)$$

It is well known, e.g., see [5], that the maximum number of errors in an RS code which can be corrected is $\lfloor (d - 1 - \nu)/2 \rfloor$ where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x, i.e., the principal part of x. Now define a generalization of the Forney syndrome polynomial.

Definition 1. The Forney syndrome polynomial is defined by

$$T(x) \equiv S(x)\,\Lambda(x) \bmod x^{d-1} \qquad (7)$$

By (7) and the key equation in (6b), $A(x)$ is

$$A(x) \equiv T(x)\,\lambda(x) \bmod x^{d-1} \qquad (8a)$$

where

$$\deg\{\lambda(x)\} \leqslant \lfloor (d - 1 - \nu)/2 \rfloor$$

and

$$\deg\{A(x)\} \leqslant \lfloor (d + \nu - 3)/2 \rfloor \qquad (8b)$$

Using a technique similar to that used for the proof of Theorem 7.7.3 in [3], one can prove an important theorem that the errata evaluator polynomial $A(x)$ and the errata locator polynomial $\tau(x)$ can be obtained simultaneously and simply from the known $T(x)$, defined above, and the new key equation in (8). This algorithm takes into account both errors and erasures.

Theorem 7.7.1 in [3] can be used to solve for $A(x)$ and $\lambda(x)$ in (8). This theorem is the classical Euclidean algorithm for polynomials in matrix form. It is restated in terms of the polynomials and notation of the present article as follows:

Theorem 1: Given the two polynomials x^{d-1} and $T(x)$ in (7), where deg $\{x^{d-1}\} >$ deg $\{T(x)\}$. Let $M_0(x) = x^{d-1}$ and $R_0(x) = T(x)$. Also, let $N_s(x)$ be the 2 X 2 matrix equation which satisfies

$$N_s(x) = \begin{bmatrix} 0 & 1 \\ 1 & -Q_{s-1}(x) \end{bmatrix} N_{s-1}(x) \qquad (9a)$$

where

$$Q_{s-1}(x) = \left\lfloor \frac{M_{s-1}(x)}{R_{s-1}(x)} \right\rfloor \qquad (9b)$$

Finally, let

$$\begin{bmatrix} M_s(x) \\ R_s(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q_{s-1}(x) \end{bmatrix} \begin{bmatrix} M_{s-1}(x) \\ R_{s-1}(x) \end{bmatrix} = N_s(x) \begin{bmatrix} x^{d-1} \\ T(x) \end{bmatrix} \qquad (9c)$$

Then, for $s = r$, $M_s(x)$ satisfies

$$M_r(x) = c \,\text{GCD}\, [x^{d-1}, T(x)]$$

where c is a scalar, and $R_r(x) = 0$.

Proof: See [3, p. 194].

By Theorem 1 one observes that

$$N_s(x) = \prod_{t=0}^{s-1} \begin{bmatrix} 0 & 1 \\ 1 & -Q_t(x) \end{bmatrix} = \begin{bmatrix} A_s(x) & B_s(x) \\ C_s(x) & D_s(x) \end{bmatrix} \qquad (10a)$$

where

$$\deg \{D_s(x)\} > \deg \{B_s(x)\} \qquad (10b)$$

It follows from (10a) that the determinant of $N_s(x)$ is $(-1)^s$. Thus, the inverse of $N_s(x)$ is given by

$$N_s^{-1}(x) = \begin{bmatrix} A_s(x) & B_s(x) \\ C_s(x) & D_s(x) \end{bmatrix}^{-1} = (-1)^s \begin{bmatrix} D_s(x) & -B_s(x) \\ -C_s(x) & A_s(x) \end{bmatrix} \qquad (10c)$$

From (10a), $D_s(x)$ satisfies the recursive equation

$$D_s(x) = D_{s-2}(x) - Q_{s-1}(x) D_{s-1}(x) \qquad (11)$$

for $s = 1, 2, \ldots,$ where the initial conditions are $D_{-1}(x) = 0$ and $D_0(x) = 1$. Also from (9c), one obtains the equations

$$M_s(x) = R_{s-1}(x) \qquad (12a)$$

$$R_s(x) = M_{s-1}(x) - Q_{s-1}(x) R_{s-1}(x) \qquad (12b)$$

for $s = 1, 2, \ldots,$ where

$$\deg \{M_s(x)\} > \deg \{R_s(x)\} \qquad (12c)$$

Thus, from (12a) and (12b), the recursive formula for $R_s(x)$ is

$$R_s(x) = R_{s-2}(x) - Q_{s-1}(x) R_{s-1}(x) \qquad (12d)$$

for $s = 1, 2, \ldots,$ where the initial conditions are $R_{-1}(x) = x^{d-1}$ and $R_0(x) = T(x)$.

A substitution of (10a) into (9c) yields

$$\begin{bmatrix} M_s(x) \\ R_s(x) \end{bmatrix} = \begin{bmatrix} A_s(x) & B_s(x) \\ C_s(x) & D_s(x) \end{bmatrix} \begin{bmatrix} x^{d-1} \\ T(x) \end{bmatrix} \qquad (13)$$

Thus, from (13), one obtains

$$R_s(x) \equiv T(x) D_s(x) \bmod x^{d-1} \qquad (14)$$

for $s = 1, 2, \ldots,$ where the recursive formulas for $D_s(x)$ and $R_s(x)$ are given in (11) and (12d), respectively.

Equation (14) is the form of Eq. (8a), which is used to solve for both $A(x)$ and $\lambda(x)$. To solve for $A(x)$ and $\lambda(x)$, one needs to find by (8b) a unique value $s = s'$, such that

$$\deg \{D_{s'}(x)\} \leqslant \lfloor (d - 1 - \nu)/2 \rfloor$$

and

$$\deg \{R_{s'}(x)\} \leqslant \lfloor((d-1-\nu)/2)-1\rfloor = \lfloor(\nu+d-3)/2\rfloor$$

in (14). Since $\deg \{R_{-1}(x)\} = d-1$ and $\deg \{R_s(x)\}$ is strictly decreasing with an increase in s. there always exists a unique value s' such that

$$\deg \{R_{s'-1}(x)\} \geqslant \left\lceil \frac{\nu+d-1}{2} \right\rceil \tag{15a}$$

and

$$\deg \{R_{s'}(x)\} \leqslant \left\lfloor \frac{\nu+d-3}{2} \right\rfloor \tag{15b}$$

where $\lceil x \rceil$ denotes the least integer greater than or equal to x.

Since $\deg \{D_{s'}(x)\}$ is increasing with an increase in s, then one needs to show also that

$$\deg \{D_{s'}(x)\} \leqslant \lfloor(d-1-\nu)/2\rfloor \tag{16}$$

To prove (16), it follows from (13) that

$$\begin{bmatrix} x^{d-1} \\ T(x) \end{bmatrix} = N_{s'}^{-1}(\lambda) \begin{bmatrix} M_{s'}(x) \\ R_{s'}(x) \end{bmatrix} \tag{17}$$

for $s = s'$, where $N_{s'}^{-1}(x)$ is the inverse of the 2×2 matrix $N_{s'}(x)$ in (10a). The substitution (10c) into (17) yields

$$\begin{bmatrix} x^{d-1} \\ T(x) \end{bmatrix} = (-1)^{s'} \begin{bmatrix} D_{s'}(x) & -B_{s'}(x) \\ -C_{s'}(x) & A_{s'}(x) \end{bmatrix} \begin{bmatrix} M_{s'}(x) \\ R_{s'}(x) \end{bmatrix} \tag{18}$$

From (18), one obtains

$$x^{d-1} = (-1)^{s'} \left[D_{s'}(x) M_{s'}(x) - B_{s'}(x) R_{s'}(x)\right] \tag{19a}$$

From (10b) and (12c), one knows that

$$\deg \{M_{s'}(x)\} > \deg \{R_{s'}(x)\}$$

and

$$\deg \{D_{s'}(x)\} > \deg \{B_{s'}(x)\}.$$

Thus, by (19a),

$$\deg \{x^{d-1}\} = \deg \{D_{s'}(x)\} + \deg \{M_{s'}(x)\} \tag{19b}$$

Since by (12a) for $s = s'$, $M_{s'}(x) = R_{s'-1}(x)$ and $\deg \{x^{d-1}\} = d-1$, then (19b) becomes

$$\deg \{D_{s'}(x)\} = d-1-\deg \{R_{s'-1}(x)\} \quad . \tag{20a}$$

By (15a), $\deg D_{s'}(x)$ in (20a) is upper bounded by the following inequality:

$$\deg \{D_{s'}(x)\} \leqslant d-1-\left\lceil \frac{\nu+d-1}{2} \right\rceil = \left\lfloor \frac{d-1-\nu}{2} \right\rfloor \tag{20b}$$

By (15b) and (20b), the Euclidean algorithm has the following stop conditions:

$$\deg \{R_{s'}(x)\} \leqslant \lfloor(\nu+d-3)/2\rfloor$$

and

$$\deg \{D_{s'}(x)\} \leqslant \lfloor(d-1-\nu)/2\rfloor$$

Thus, also by (15a) and (20b), polynomials $R_{s'}(x)$ and $D_{s'}(x)$ are solutions of (14). Uniqueness follows from the fact that there is only one solution to (8a) under the conditions of (8b). Therefore, $A(x) = R_{s'}(x)/\Delta$ and $\lambda(x) = D_{s'}(x)/\Delta$ are the unique solutions of (8a, b), where Δ is chosen to ensure that $\lambda_0 = 1$.

To obtain $\tau(x)$, one replaces the recursive equation in (11) by

$$\tau_s(x) = -Q_{s-1}(x)\,\tau_{s-1}(x) + \tau_{s-2}(x) \tag{21}$$

with the changed initial conditions: $\tau_0(x) = \Lambda(x)$ and $\tau_{-1}(x) = 0$. To obtain the final errata locator polynomial, first, let $s = 1$. Then (21) becomes by (11),

$$\begin{aligned} \tau_1(x) &= -Q_0(x)\,\tau_0(x) + \tau_{-1}(x) \\ &= -Q_0(x)\,\Lambda(x) = D_1(x)\,\Lambda(x) \end{aligned}$$

For $s = 2$, (21) becomes by (11),

$$\begin{aligned} \tau_2(x) &= -Q_1(x)\,\tau_1(x) + \tau_0(x) = -Q_1(x)\,D_1(x)\,\Lambda(x) + \Lambda(x) \\ &= (-Q_1(x)\,D_1(x) + 1)\,\Lambda(x) = D_2(x)\,\Lambda(x) \end{aligned}$$

etc. Here, in general, one has $\tau_s(x) = D_s(x)\,\Lambda(x)$ with $\tau_0(x) = \Lambda(x)$ and $\tau_{-1}(x) = 0$.

The results proved above imply the following now evident theorem:

Theorem 2: Let $T(x)$ in (7) be the Forney syndrome polynomial of a t-error and ν-erasure correcting RS code under the condition $\nu + 2t \leqslant d - 1$, where $d - 1$ is either an even or an odd integer. Consider the two polynomials x^{d-1} and $T(x)$ in (7). The Euclidean algorithm for polynomials on $GF(2^m)$ can be used to develop two finite sequences $R_s(x)$ and $\tau_s(x)$ from the following two recursive formulas:

$$\tau_s(x) = (-Q_{s-1}(x))\,\tau_{s-1}(x) + \tau_{s-2}(x) \tag{22a}$$

and

$$R_s(x) = R_{s-2}(x) - Q_{s-1}(x) R_{s-1}(x) \tag{22b}$$

for $s = 1, 2, \ldots$, where the initial conditions are $\tau_0(x) = \Lambda(x)$, $\tau_{-1}(x) = 0$, $R_{-1}(x) = x^{d-1}$, and $R_0(x) = T(x)$. Here $Q_{s-1}(x)$ is obtained as the principal part of $R_{s-2}(x)/R_{s-1}(x)$. The recursion in (22a) and (22b) for $R_s(t)$ and $\tau_s(x)$ teiminates when deg $\{R_s(x)\} \leqslant \lfloor (d + \nu - 3)/2 \rfloor$ for the first time for some value $s = s'$. Let

$$A(x) = R_{s'}(x)/\Delta \tag{23a}$$

and

$$\tau(x) = \tau_{s'}(x)/\Delta \tag{23b}$$

Also in (23) $\Delta = \tau_{s'}(0)$ is a field element in $GF(2^m)$ which is chosen so that $\tau_0 = 1$. Then $A(x)$ and $\tau(x)$ in (23) are the unique solution of $A(x) \equiv T(x)\,\tau(x) \bmod x^{d-1}$, where both inequalities, deg $\{\tau(x)\} \leqslant \lfloor (d + \nu - 1)/2 \rfloor$ and deg $\{A(x)\} \leqslant \lfloor (d + \nu - 3)/2 \rfloor$, are satisfied.

The proof of Theorem 2 demonstrates that the algorithm presented by Eastman[1] is correct.

The roots of $\tau(x)$ are the inverse locations of the t errors and ν erasures. These roots are most efficiently found by the Chien search procedure. By (4d) it is shown readily that the errata values are

$$Y_k = \frac{A(X_k^{-1})}{(X_k^{b-1}\,\tau'(X_k^{-1}))} \qquad \text{for } 1 \leqslant k \leqslant \nu + t \tag{24}$$

where $\tau'(X_k^{-1})$ is the derivative with respect to x of $\tau(x)$, evaluated at $x = X_k^{-1}$.

The overall time-domain decoding of RS codes for correcting errors and erasures, using Theorem 2 and the Euclidean algorithm, is summarized in the following steps:

(1) Compute the transform of the received vector m-tuple over $GF(2^m)$ from Eq. (2). Next calculate the erasure locator polynomial $\Lambda(x)$ from Eq. (4a) and define deg $\{\Lambda(x)\} = \nu$.

(2) Compute the Forney syndrome polynomial from $T(x)$ in (7).

(3) To determine the errata locator polynomial $\tau(x)$ and errata evaluator polynomial $A(x)$, where $0 \leqslant \nu < d - 1$, apply the Euclidean algorithm to x^{d-1} and $T(x)$ as given by (7). The initial values of the Euclidean algorithm are $\tau_0(x) = \Lambda(x)$, $\tau_{-1}(x) = 0$, $\mathrm{R}_{-1}(x) = x^{d-1}$ and $R_0(x) = T(x)$. For $\nu = d - 1$ set $\tau(x) = \Lambda(x)$ and $A(x) = T(x)$.

(4) Compute the errata values from (24).

To illustrate the time domain decoding procedure for correcting errors and erasures, an elementary example of an RS code over $GF(2^4)$ is now presented. The representation of the field $GF(2^4)$ generated by the primitive irreducible polynomial $G(x) = x^4 + x + 1$ is given in Table A-1 in the appendix.

Example 1: Consider a (15, 9) RS code over $GF(2^4)$ with minimum distance $d = 7$. In this code, ν erasures and t errors under the condition $2t + \nu \leqslant d - 1$ can be corrected. In order to simplify this example, let $\gamma = \alpha$ and $b = 1$. Thus, the generator polynomial of such a (15, 9) RS code is defined by

$$g(x) = \prod_{i=1}^{6} (x - \alpha^i) = x^6 + \alpha^{10}x^5 + \alpha^{14}x^4$$

$$+ \alpha^4 x^3 + \alpha^6 x^2 + \alpha^9 x + \alpha^6$$

Assume the message symbols are

$$I(x) = \alpha^{10}x^{14} + \alpha^{12}x^{13} + \alpha^8 x^{12} + \alpha^5 x^{11}$$

$$+ \alpha^6 x^{10} + \alpha^{14}x^9 + \alpha^{13}x^8 + \alpha^{11}x^7 + \alpha^9 x^6$$

The encoded codeword, which is a multiple of $g(x)$, is

$$c(x) = \alpha^{10}x^{14} + \alpha^{12}x^{13} + \alpha^8 x^{12} + \alpha^5 x^{11} + \alpha^6 x^{10}$$

$$+ \alpha^{14}x^9 + \alpha^{13}x^8 + \alpha^{11}x^7 + \alpha^9 x^6 + x^5 + \alpha x^4$$

$$+ \alpha^2 x^3 + \alpha^6 x^2 + \alpha^{12}x + \alpha^8$$

Written as a vector, the codeword is

$$c = (\alpha^{10}, \alpha^{12}, \alpha^{8}, \alpha^{5}, \alpha^{6}, \alpha^{14}, \alpha^{13}, \alpha^{11}, \alpha^{9}, \alpha^{0}, \alpha,$$

$$\alpha^{2}, \alpha^{6}, \alpha^{12}, \alpha^{8})$$

Assume the erasure vector is

$$u = (0, 0, 0, 0, 0, 0, 0, \alpha^2, 0, 0, 0, 0, 0, 0, 0)$$

and the error vector is

$$e = (0, 0, 0, 0, \alpha^{11}, 0, 0, 0, 0, 0, 0, \alpha^7, 0, 0, 0)$$

Then the errata vector is

$$\tilde{u} = u + e = (0, 0, 0, 0, \alpha^{11}, 0, 0, \alpha^2, 0, 0, 0, \alpha^7, 0, 0, 0)$$

Assume the received vector is

$$r = c + \tilde{u} = (\alpha^{10}, \alpha^{12}, \alpha^{8}, \alpha^{5}, \alpha, \alpha^{14}, \alpha^{13}, \alpha^{9}, \alpha^{9}, \alpha^{0}, \alpha,$$

$$\alpha^{12}, \alpha^{6}, \alpha^{12}, \alpha^{8})$$

The syndromes S_k for r are

$$S_k = \sum_{n=0}^{14} r_n \alpha^{n \cdot k} = \alpha^7 (\alpha^3)^k + \alpha^2 (\alpha^7)^k + \alpha^{11} (\alpha^{10})^k \quad \text{for } 1 \leqslant k \leqslant 6$$

This yields $S_1 = \alpha^0$, $S_2 = \alpha^{13}$, $S_3 = \alpha^{14}$, $S_4 = \alpha^{11}$, $S_5 = \alpha$, and $S_6 = 0$. Thus, the syndrome polynomial is

$$S(x) = \alpha^0 + \alpha^{13} x + \alpha^{14} x^2 + \alpha^{11} x^3 + \alpha x^4 + 0 x^5$$

The erasure locator polynomial is $\Lambda(x) = (1 + \alpha^7 x)$. In this example, the maximum erasure correcting capability is

$$\lfloor (d - 1 - \nu)/2 \rfloor = \lfloor (7 - 1 - 1)/2 \rfloor = 2$$

By (7), one obtains the Forney syndrome polynomial as

$$T(x) \equiv \Lambda(x) S(x) \equiv (1 + \alpha^7 x)(1 + \alpha^{13} x + \alpha^{14} x^2 + \alpha^{11} x^3$$

$$+ \alpha x^4 + 0 x^5) \bmod x^6$$

$$= \alpha^8 x^5 + \alpha^9 x^4 + \alpha x^3 + \alpha^{12} x^2 + \alpha^5 x + \alpha^0 \tag{25}$$

In (25), the coefficients of $T(x)$, $T_0 = \alpha^0$, $T_1 = \alpha^5$, $T_2 = \alpha^{12}$, $T_3 = \alpha$, $T_4 = \alpha^9$, and $T_5 = \alpha^8$ are the Forney syndromes.

The Euclidean algorithm is applied next to polynomials x^{d-1} and $T(x)$ in (7). By this means, polynomials $\tau(x)$ and $A(x)$ are determined by use of the Euclidean algorithm. This is accomplished by the recursive formulas (22a) and (22b) illustrated in Table 1, where initially $R_{-1}(x) = x^{d-1} = x^6$ and $R_0(x) = T(x) = \alpha^8 x^5 + \alpha^9 x^4 + \alpha x^3 + \alpha^{12} x^2 + \alpha^5 x + 1$. From Table 1, one observes that deg $\{R_{s'}(x)\}$ = deg $\{R_2(x)\} = 2 \leqslant \lfloor (d + \nu - 3)/2 \rfloor = 2$. Thus, the computation terminates at this point for $s' = 2$, and

$$R_2(x) = \alpha^7 x^2 + \alpha x + \alpha^2$$

and

$$\tau_2(x) = \alpha^7 x^3 + \alpha^{13} x^2 + \alpha^4 x + \alpha^2$$

By (23a) and (23b), one has

$$\tau(x) = \frac{1}{\alpha^2} \tau_2(x) = \alpha^5 x^3 + \alpha^{11} x^2 + \alpha^2 x + 1 \tag{26}$$

and

$$A(x) = \frac{1}{\alpha^2} R_2(x) = \alpha^5 x^2 + \alpha^{14} x + 1 \tag{27}$$

By use of a Chien search, the roots of $\tau(x)$ constitute the set $\{\alpha^{-7}, \alpha^{-3}, \alpha^{-10}\}$. The derivative with respect to x of $\tau(x)$ in (26) is $\tau'(x) = \alpha^5 x^2 + \alpha^2$. Thus, by (24) and (27), the errata values are

$$Y_1 = \frac{A(X_1^{-1})}{\tau'(X_1^{-1})} = \frac{A(\alpha^{-7})}{\tau'(\alpha^{-7})} = \frac{\alpha^5 (\alpha^{-7})^2 + \alpha^{14} (\alpha^{-7}) + 1}{\alpha^5 (\alpha^{-7})^2 + \alpha^2} = \alpha^2$$

$$Y_2 = \frac{A(X_2^{-1})}{\tau'(X_2^{-1})} = \frac{\alpha^5 (\alpha^{-3}) + \alpha^{14} (\alpha^{-3}) + 1}{\alpha^5 (\alpha^{-3})^2 + \alpha^2} = \alpha^7$$

and

$$Y_3 = \frac{A(X_3^{-1})}{\tau'(X_3^{-1})} = \frac{\alpha^5 (\alpha^{-10})^2 + \alpha^{14} (\alpha^{-10}) + 1}{\alpha^5 (\alpha^{-10})^2 + \alpha^2} = \alpha^{11}$$

III. An Improved Architecture for the Time Domain Decoder

An improved VLSI architecture of a pipeline decoder for correcting both errors and erasures of a (255, 223) RS decoder for codes over $GF(2^8)$ is presented in Fig. 2. For this example, assume $\gamma = \alpha$ and $b = 112$.

In Fig. 2, the block diagram can be separated into two parts as indicated by the dashed line. The functional units contained in the first part of the decoder architectures in Fig. 2 are shown as follows: (1) the syndrome computation unit, (2) the power calculation unit, (3) the power expansion unit, (4) the polynomial expansion unit, (5) the $\lfloor(d + \nu - 3)/2\rfloor$ generator, and (6) the delay unit.

The syndrome computation unit accepts received messages and computes their syndromes. There are 32 syndrome subcells in a (255, 223) RS decoder. The computed syndrome polynomial is labelled as $S(x)$ in Fig. 2. The coefficients of $S(x)$ are fed in parallel to the polynomial expansion unit in order to compute the Forney syndromes.

The power calculation unit converts the received 1's and 0's into a sequence of α^k's and 0's, where α is a primitive element of the finite field over which the RS code is defined. These received 1's and 0's indicate the occurrence or nonoccurrence, respectively, of an erasure at a specific location. Since the maximum erasure correcting capability of a (255, 223) RS decoder is 32, only 32 symbol latches are needed to store all correctable erasure locations.

A detection circuit for detecting the occurrence of erasures is included in the power calculation unit. If an erasure occurs at the kth location, a symbol α^k is calculated by the power calculation unit and latched. The sequence of α^k's is fed to the polynomial expansion circuit, to the power expansion unit and to the $\lfloor(d + \nu - 3)/2\rfloor$ generator.

The power expansion unit converts the α^k's into an erasure locator polynomial $\Lambda(x)$. Therefore, the polynomial $\Lambda(x)$ has α^k's as its roots. The erasure locator polynomial $\Lambda(x)$ is fed to the modified GCD unit as one of the initial conditions for the modified GCD unit.

A generator is used to compute $\lfloor(d + \nu - 3)/2\rfloor$. The output is sent to the modified GCD unit and used as a stop indicator for the Euclidean algorithm. The polynomial expansion unit is used to compute the required Forney syndromes.

In Fig. 2, the erasure locator polynomial $\Lambda(x)$, together with the Forney syndrome polynomial $T(x)$, is the input to the modified GCD unit. The outputs of the modified GCD unit are the errata locator polynomial, $\tau(x)$, and the errata evaluator polynomial, $A(x)$. The error correcting capability of the code is computed by $\lfloor(32 - \nu)/2\rfloor$.

The second half of Fig. 2 is labelled as "II." One of the outputs of the modified GCD unit is the errata locator polynomial $\tau(x)$. This output is fed to a Chien search unit and to another unit for computing $[x^{b-1}\tau'(x)]^{-1} = [x^{111}\tau'(x)]^{-1}$, where $b = 112$. The other output of the modified GCD is the errata evaluator polynomial $A(x)$. This is fed to the polynomial evaluation unit to perform the evaluation of $A(x)$.

The $[x^{111}\tau'(x)]^{-1}$ unit computes one part of the errata magnitude. The product of the outputs from the polynomial evaluation unit and the $[x^{111}\tau'(x)]^{-1}$ unit forms the errata magnitude.

In Fig. 2, the Chien search unit is used to search for both the error and erasure locations. The architecture of the Chien search unit is similar to that of a polynomial evaluation unit, except that there is a zero detector at the end in the Chien search unit.

Compared with the previous design in Fig. 1, one observes that this improved architecture does not require the polynomial multiplication unit, delay II, delay III, and the truncation circuit usually needed in Fig. 1 for computing the errata evaluator polynomial. Thus, this new decoding algorithm in Fig. 2 is simpler and more suitable for the VLSI implementation. Finally, a comparison of VLSI architecture for time and transform domain decoding of Reed-Solomon codes is shown in [7].

References

[1] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Inf. and Contr.*, Vol. 27, pp. 87–99, 1975.

[2] G. D. Forney, "On decoding BCH codes," *IEEE Trans. on Information Theory*, Vol. IT–11, pp. 549–557, 1965.

[3] R. E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Co., CA, p. 258, May 1984.

[4] I. S. Reed, T. K. Truong, and R. L. Miller, "Decoding of B.C.H. and RS codes with errors and erasures using continued fractions," *Electronics Letters*, Vol. 15, No. 17, pp. 542–544, August 16, 1976.

[5] E. P. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, 1968.

[6] E. R. Berlekamp, "Bit-Serial Reed-Solomon Encoders," *IEEE Trans. on Information Theory*, Vol. IT–28, No. 6, pp. 869–874, November 1982.

[7] T. K. Truong, I. S. Hsu, I. S. Reed, E. Satorius and L. J. Deutsch, "A Comparison of VLSI architecture for time and transform domain decoding of Reed-Solomon codes," presented at ICCD '87 Conference, New York, October 5-8, 1987.

Table 1. An example of the Euclidean algorithm used to find $\tau(x)$ and $A(x)$

S	$R_{s-2}(x) = Q_{s-1}(x)R_{s-1}(x) + R_s(x)$	$Q_{s-1}(x)$	$R_s(x)$	$\tau_s(x)$
-1			x^6	0
0			$\alpha^8x^5 + \alpha^9x^4 + \alpha x^3$ $+ \alpha^{12}x^2 + \alpha^5x + 1$	$1 + \alpha^7x$
1	$\frac{1}{\alpha^8}x + \frac{\alpha}{\alpha^8}$ $\alpha^8x^5 + \alpha^9x^4 + \alpha x^3 + \alpha^{12}x^2 + \alpha^5x + 1 \,)\, \overline{x^6}$ $\underline{x^6 + \alpha x^5 + \alpha^8x^4 + \alpha^4x^3 + \alpha^{12}x^2 + \alpha^7 x}$ $\alpha x^5 + \alpha^8x^4 + \alpha^4x^3 + \alpha^{12}x^2 + \alpha^7x$ $\underline{\alpha x^5 + \alpha^2x^4 + \alpha^9x^3 + \alpha^5x^2 + \alpha^{13}x + \alpha^8}$ $x^4 + \alpha^{14}x^3 + \alpha^{14}x^2 + \alpha^3x + \alpha^8$	$\frac{1}{\alpha^8}(x + \alpha)$	$x^4 + \alpha^{14}x^3 + \alpha^6x^2$ $+ \alpha^2x + \alpha^8$	$\frac{1}{\alpha^8}(x + \alpha)$ $\cdot (1 + \alpha^7x) + 0$ $= \frac{1}{\alpha^8}(\alpha^7x^2$ $+ \alpha^2x + \alpha)$
2	$\alpha^8x + 1$ $x^4 + \alpha^{14}x^3 + \alpha^{14}x^2 + \alpha^5x + \alpha^8 \,)\, \overline{\alpha^8x^5 + \alpha^9x^4 + \alpha x^3 + \alpha^{12}x^2 + \alpha^5x + 1}$ $\underline{\alpha^8x^5 + \alpha^7x^4 + \alpha^7x^3 + \alpha^{13}x^2 + \alpha x}$ $x^4 + \alpha^{14}x^3 + \alpha x^2 + \alpha^2x + 1$ $\underline{x^4 + \alpha^{14}x^3 + \alpha^{14}x^2 + \alpha^5x + \alpha^8}$ $\alpha^7x^2 + \alpha x + \alpha^2$	$\alpha^8x + 1$	$\alpha^7x^2 + \alpha x + \alpha^2$	$(\alpha^7x^2 + \alpha^2x + \alpha)$ $\cdot (x + \alpha^7) + (1 + \alpha^7x)$ $= \alpha^7x^3 + \alpha^{13}x^2$ $+ \alpha^4x + \alpha^2$

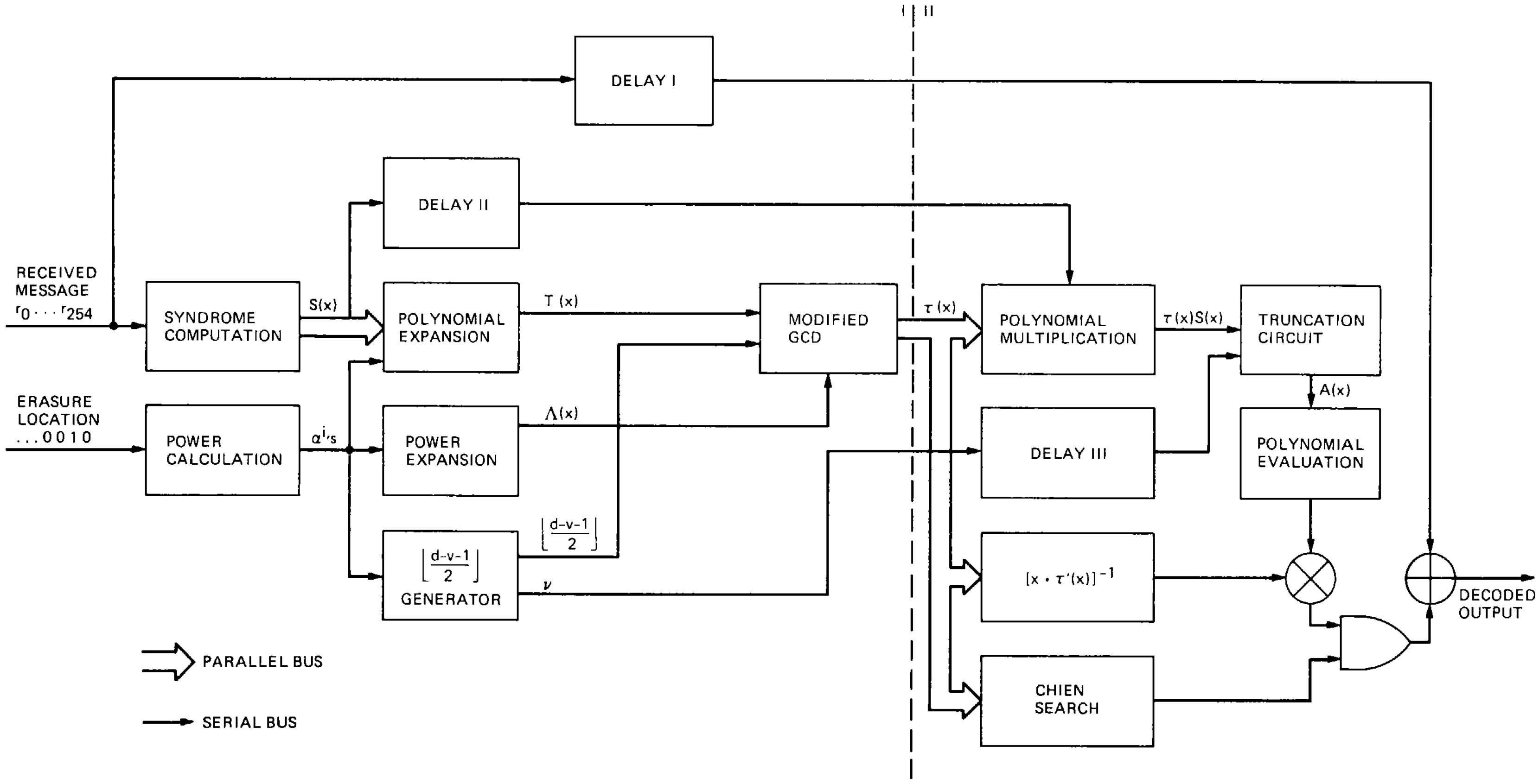

Fig. 1. An unmodified block diagram of a pipeline (255,223) RS time domain decoder

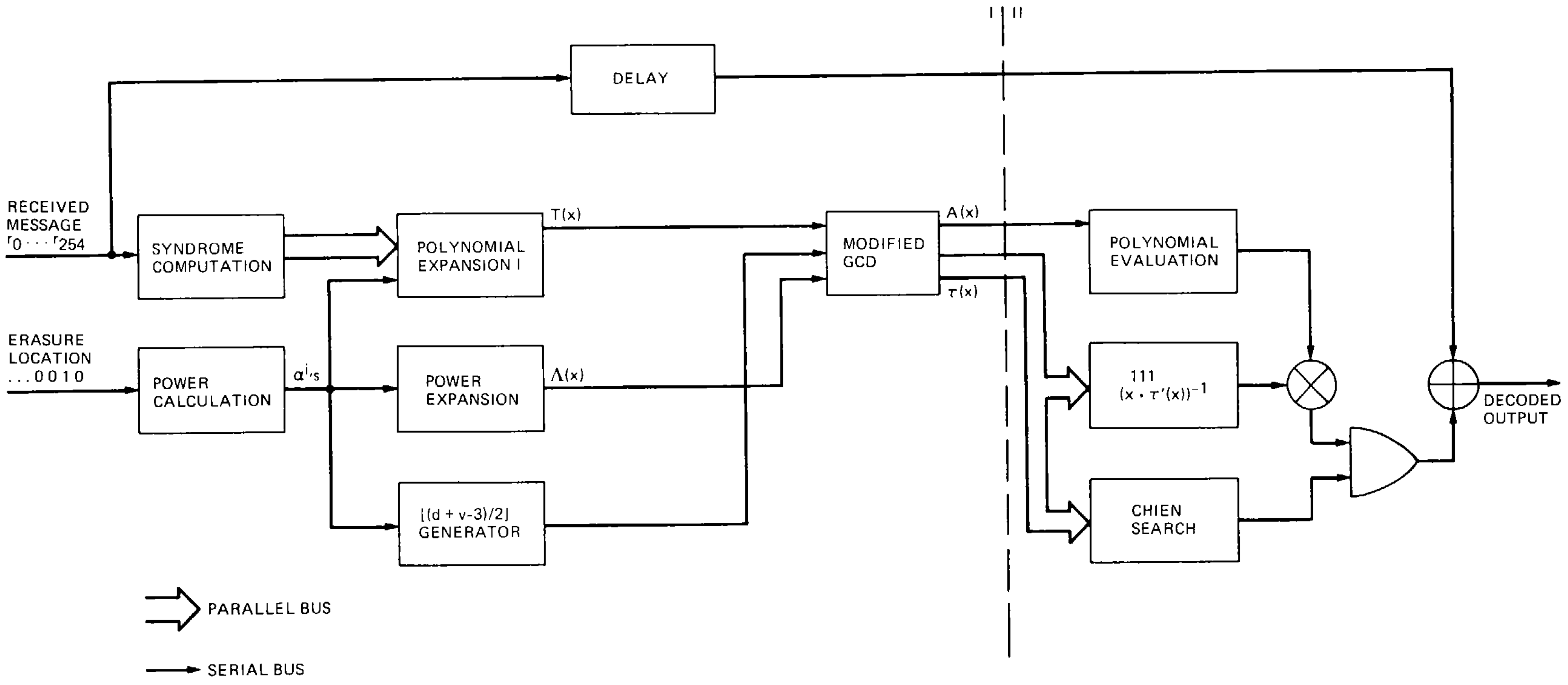

Fig. 2. A modified block diagram of a pipeline (255,223) RS time domain decoder

Appendix

Table A-1. Representations of the elements of $GF(2^4)$ generated by $\alpha^4 + \alpha + 1 = 0$

	α^3	α^2	α	α^0
α^0	0	0	0	1
α^1	0	0	1	0
α^2	0	1	0	0
α^3	1	0	0	0
α^4	0	0	1	1
α^5	0	1	1	0
α^6	1	1	0	0
α^7	1	0	1	1
α^8	0	1	0	1
α^9	1	0	1	0
α^{10}	0	1	1	1
α^{11}	1	1	1	0
α^{12}	1	1	1	1
α^{13}	1	1	0	1
α^{14}	1	0	0	1