

A Comparison of VLSI Architectures for Time and Transform Domain Decoding of Reed–Solomon Codes

I. S. Hsu, T. K. Truong, L. J. Deutsch, and E. H. Satorius
Communications Systems Research Section

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 needed to decode a Reed–Solomon (RS) code. It is shown in this article that this algorithm can be used for both time and transform domain decoding by replacing its initial conditions with the Forney syndromes and the erasure locator polynomial. By this means both the error locator polynomial and the erasure evaluator polynomial can be obtained with the Euclidean algorithm.

With these ideas, both time and transform domain Reed–Solomon decoders for correcting errors and erasures are simplified and compared in this article. As a consequence, the architectures of Reed–Solomon decoders for correcting both errors and erasures can be made more modular, regular, simple, and naturally suitable for VLSI implementation.

I. Introduction

The Euclidean algorithm for solving the key equation for decoding Bose–Chaudhuri–Hocquenghem (BCH) and Goppa type codes was first developed by Sugiyama *et al.* [1]. The authors [2], [3] derived a fast decoding of Reed–Solomon (RS) codes using the continued fraction, which is closely related to the Euclidean algorithm. Brent and Kung [4] were the first to suggest a systolic array architecture for computing the greatest common divisor (GCD) of two polynomials. Through the use of these ideas, a pipeline structure for a transform domain decoder was developed to decode errors of RS codes [5]. An important ingredient of this design is a modi-

fied Euclidean algorithm for computing the error locator polynomial.

The computation of inverse field elements is completely avoided in the above-mentioned modification of Euclid's algorithm. Recently, the authors [6] proposed that a recursive algorithm could be used to perform this modified Euclidean algorithm. An important advantage of this new recursive algorithm is that the entire systolic array needed to perform Euclid's algorithm requires substantially less silicon area than the pipeline version of the modified Euclidean algorithm given in [5].

Forney [13] defined an errata locator polynomial using what are now called Forney syndromes to correct both errors and erasures. Blahut [7] showed that the errata locator polynomial can be computed directly by initializing Berlekamp's algorithm with the erasure locator polynomial.

Recently Eastman [8] suggested that the errata evaluator polynomial can be computed directly by initializing Berlekamp's algorithm with the Forney syndrome polynomial. This new, simplified decoding procedure is proved in [9]. By this technique, it is possible to compute the errata locator polynomial and the errata evaluator polynomial simultaneously from the Euclidean algorithm. This new RS decoder uses both the erasure locator polynomial and the Forney syndrome polynomial as initial conditions for the Euclidean algorithm.

It is shown and proved in [9] that the modified Euclidean algorithm mentioned above can be used to solve the Berlekamp-Massey key equation for the errata locator polynomial and the errata evaluator polynomial directly and simultaneously. By this means a new, simplified pipeline architecture for both the time and transform domain decoders can be developed for correcting both errors and erasures of RS codes. Such a decoding technique can be faster and simpler than previous methods [15], [10].

In this article, it is found that the VLSI implementation of the transform domain decoder is simpler than that of the time domain decoder. However, for a long RS code (10 bits or larger), due to the large size of the inverse transform unit needed in the transform decoder, the VLSI area needed to implement the transform domain decoder can be substantially larger than that needed for the time domain decoder. For moderately long codes, such as the 8-bit (255, 223) RS code used in the concatenated coding system for NASA's Voyager mission [11], the transform domain decoder is still simpler than the time domain decoder.

The above-mentioned NASA coding system is called the "baseline" system. It uses a (7, 1/2) convolutional code as its inner code and an 8-bit (255, 223) RS code as its outer code. It is shown [12] that this system achieves a bit-error rate (BER) of 10^{-6} at a bit signal-to-noise ratio (SNR) of 2.53 dB.

As mentioned above, the time domain decoder is more efficient in area than the transform domain decoder for very long RS codes. One such example is the long, 10 bits/symbol (1023, 959) RS code presently being considered for very deep space probes. If this code is concatenated with a (15, 1/5) convolutional code, it achieves a BER of 10^{-6} at an SNR of 0.5 dB [12]. Evidently the new NASA concatenated coding system provides a 2 dB improvement over the present baseline

system. It is for this reason and many other applications that it is important to develop an efficient, VLSI implementable, time domain RS decoder.

II. The Time Domain Decoder for RS Codes

An algorithm is developed in [15] for time domain decoding of RS codes to correct both errors and erasures through the use of continued fractions or their equivalent, Euclid's algorithm. This algorithm is a modification of the Berlekamp-Forney method [13], [14]. In this algorithm, the continued fraction algorithm is used to find the error locator polynomial from the remainder of the formal power series for the Forney syndrome. The disadvantage of this algorithm is that after the error locator polynomial is obtained by continued fractions, two polynomial multiplications are needed to compute the errata locator polynomial and the errata evaluator polynomial from the known error locator polynomial.

In this section, the above-mentioned algorithm is modified to correct both errors and erasures in the time domain decoding of RS codes by the use of the Euclidean algorithm. In this new algorithm, 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 [15] is that separate computation of the errata locator polynomial and the errata evaluator polynomial, which is usually needed [15], can be avoided. This new decoding algorithm is highly suitable for 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. The following five vectors are defined as:

$$\begin{aligned} \mathbf{c} &= (c_0, c_1, \dots, c_{N-1}), \text{ code vector} \\ \mathbf{r} &= (r_0, r_1, \dots, r_{N-1}), \text{ received vector} \\ \mathbf{e} &= (e_0, e_1, \dots, e_{N-1}), \text{ error vector} \\ \mathbf{u} &= (u_0, u_1, \dots, u_{N-1}), \text{ erasure vector} \\ \bar{\mathbf{u}} &= (\bar{u}_0, \bar{u}_1, \dots, \bar{u}_{N-1}), \text{ errata vector} \end{aligned}$$

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

Suppose that t errors and v erasures occur in the received vector \mathbf{r} , and assume that $v + 2t \leq d - 1$. 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 [16] that the generator polynomial $g(x)$ is symmetric if and only if

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

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

$$\begin{aligned} S_{(b-1)+k} &= \sum_{i=0}^{N-1} \bar{u}_i \gamma^{i(b-1+k)} = \sum_{i=0}^{N-1} (u_i + e_i) \gamma^{i(b-1+k)} \\ &= \sum_{j=1}^{v+t} Y_j X_j^{(b-1)+k} \quad \text{for } 1 \leq k \leq d-1 \end{aligned} \quad (2)$$

where X_j is either the j th erasure or error location, and Y_j is either the j th erasure or the error magnitude. Define the set $\Lambda = \{X_i | X_i \text{ is an erasure location}\}$ and $\lambda = \{X_i | X_i \text{ is an error location}\}$. Define the syndrome polynomial

$$S(x) = \sum_{k=1}^{d-1} S_{(b-1)+k} x^{k-1} \quad (3a)$$

Then it is not difficult to show (see [14]) that

$$\begin{aligned} S(x) &= \sum_{k=1}^{d-1} S_{(b-1)+k} x^{k-1} = \sum_{j=1}^{v+t} \frac{Y_j X_j^b}{(1 - X_j x)} \\ &\quad - \sum_{j=1}^{v+t} \frac{Y_j X_j x^{b+d-1}}{(1 - X_j x)} \end{aligned} \quad (3b)$$

Following [14], we define four different polynomials as follows:

The erasure locator:

$$\Lambda(x) = \prod_{X_j \in \Lambda} (1 - X_j x) = \sum_{j=1}^v (1 - X_j x) = \sum_{j=0}^v (-1)^j \Lambda_j x^j \quad (4a)$$

where $\Lambda_0 = 1$.

The error locator:

$$\lambda(x) = \prod_{X_j \in \lambda} (1 - X_j x) = \sum_{j=1}^t (1 - X_j x) = \sum_{j=0}^t (-1)^j \lambda_j x^j \quad (4b)$$

where $\lambda_0 = 1$.

The errata locator:

$$\tau(x) = \Lambda(x)\lambda(x) = \prod_{j=1}^{v+t} (1 - X_j x) = \sum_{j=0}^{v+t} (-1)^j \tau_j x^j \quad (4c)$$

where $\tau_0 = 1$.

The errata evaluator:

$$A(x) = \sum_{j=1}^{v+t} Y_j X_j^b \left(\prod_{i \neq j} (1 - X_i x) \right) \quad (4d)$$

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

$$S(x)\tau(x) = A(x) + x^{d-1} \sum_{j=1}^{v+t} Y_j X_j^{b+d-1} \left(\prod_{i \neq j} (1 - X_i x) \right) \quad (5)$$

From Eq. (5), one obtains the congruence relation,

$$S(x)\tau(x) \equiv A(x) \pmod{x^{d-1}} \quad (6a)$$

It is shown [9] that Eq. (6a) can be solved to yield

$$S(x) \equiv \frac{A(x)}{\lambda(x)\Lambda(x)} \pmod{x^{d-1}} \quad (6b)$$

It is well known, e.g., see [15], that the maximum number of errors in an RS code which can be corrected is $\lfloor (d-1-v)/2 \rfloor$ where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x , i.e., the principal part of x . We now define the Forney syndrome polynomial.

Definition 1: The Forney syndrome polynomial is defined by

$$T(x) \equiv S(x)\Lambda(x) \pmod{x^{d-1}} \quad (7)$$

By Eq. (7), the key in Eq. (6b) for $\lambda(x)$ and $A(x)$ is:

$$T(x) \equiv \frac{A(x)}{\lambda(x)} \pmod{x^{d-1}} \quad (8)$$

where

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

and

$$\deg \{A(x)\} \leq \lfloor (d+\nu-3)/2 \rfloor$$

It is shown in the following theorem that the errata evaluator polynomial $A(x)$ and the errata locator polynomial $\tau(x)$ can be obtained simultaneously and simply from knowing $T(x)$ in Eq. (7) and the new key equation in Eq. (8), which takes into account both errors and erasures.

Theorem 1: Let $T(x)$ in Eq. (7) be the Forney syndrome polynomial of a t -error and ν -erasure correcting RS code under the condition $\nu + 2t \leq d - 1$ where $d - 1$ is either an even or an odd integer. Consider the two polynomials $M(x) = x^{d-1}$ and $T(x) \equiv S(x) \Lambda(x) \pmod{x^{d-1}}$. Then 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:

$$\tau_s(x) = (-q_{s-1}(x)) \tau_{s-1}(x) + \tau_{s-2}(x) \quad (9a)$$

and

$$R_s(x) = R_{s-2}(x) - q_{s-1}(x) R_{s-1}(x) \quad (9b)$$

for $(s = 1, 2, \dots)$, where the initial conditions are $\tau_0(x) = \Lambda(x)$, $\tau_{-1}(x) = 0$, $R_{-1}(x) = M(x)$, 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 Eq. (9) for $R_s(x)$ and $\tau_s(x)$ terminates when $\deg \{R_s(x)\} \leq \nu + w - 1$ for the first time for some value $s = s'$. Let

$$A(x) = \frac{R_{s'}(x)}{\Delta} \quad (10a)$$

and

$$\tau(x) = \frac{\tau_{s'}(x)}{\Delta} \quad (10b)$$

Also in Eq. (10), $\Delta = \tau_{s'}(0)$ is a field element in $GF(2^m)$ which is chosen so that $\tau_0 = 1$. Then $A(x)$ and $\tau(x)$ in Eq. (10) are the unique solutions of

$$A(x) \equiv T(x) \tau(x) \pmod{x^{d-1}} \quad (10c)$$

where both the inequalities, $\deg \{\tau(x)\} \leq \lfloor (d+\nu-1)/2 \rfloor$ and $\deg \{A(x)\} \leq \lfloor (d+\nu-3)/2 \rfloor$, are satisfied.

Proof: Theorem 1 is a proof [9] that the idea in [8] 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 Eq. (4d) it is readily shown that the errata values are

$$Y_k = \frac{A(X_k)^{-1}}{(X_k^{b-1} \tau' X_k^{-1})} \quad \text{for } 1 \leq k \leq \nu + t \quad (11)$$

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 the Euclidean algorithm is summarized in the following steps:

- (1) Compute the transform of the received m -tuple vector 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 Eq. (7).
- (3) Determine the errata locator polynomial $\tau(x)$ and errata evaluator polynomial $A(x)$, where $0 \leq \nu < d - 1$, by applying the Euclidean algorithm to x^{d-1} and $T(x)$ as given by Eq. (7). The initial values of the Euclidean algorithm are $\tau_0(x) = \Lambda(x)$, $\tau_{-1}(x) = 0$, $R_{-1}(x) = x^{d-1}$, and $R_0(x) = T(x)$. The recursion in Eq. (9) for $R_s(x)$ and $\tau_s(x)$ terminates when $\deg \{R_s(x)\} \leq \lfloor (d+\nu-3)/2 \rfloor$ for the first time for some value $s = s'$. Finally, compute $\tau(x)$ and $A(x)$ from Eq. (10). For $\nu = d - 1$, set $\tau(x) = \Lambda(x)$ and $A(x) = T(x)$.
- (4) Compute the errata values from Eq. (11).

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 Appendix A.

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 $2t + \nu \leq d - 1$ 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

$$g(x) = \prod_{i=1}^6 (x - \alpha^i) = x^6 + \alpha^{10}x^5 + \alpha^{14}x^4 + \alpha^4x^3 + \alpha^6x^2 + \alpha^9x + \alpha^6$$

Assume the message symbols are

$$I(x) = \alpha^{10}x^{14} + \alpha^{12}x^{13} + \alpha^8x^{12} + \alpha^5x^{11} + \alpha^6x^{10} + \alpha^{14}x^9 + \alpha^{13}x^8 + \alpha^{11}x^7 + \alpha^9x^6$$

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

$$c(x) = \alpha^{10}x^{14} + \alpha^{12}x^{13} + \alpha^8x^{12} + \alpha^5x^{11} + \alpha^6x^{10} + \alpha^{14}x^9 + \alpha^{13}x^8 + \alpha^{11}x^7 + \alpha^9x^6 + x^5 + \alpha x^4 + \alpha^2x^3 + \alpha^6x^2 + \alpha^{12}x + \alpha^8$$

Written as a vector, the code word is

$$\mathbf{c} = (\alpha^{10}, \alpha^{12}, \alpha^8, \alpha^5, \alpha^6, \alpha^{14}, \alpha^{13}, \alpha^{11}, \alpha^9, \alpha^0, \alpha, \alpha^2, \alpha^6, \alpha^{12}, \alpha^8)$$

Assume the erasure vector is

$$\mathbf{u} = (0, 0, 0, 0, 0, 0, 0, \alpha^2, 0, 0, 0, 0, 0, 0, 0) \quad (12)$$

and the error vector is

$$\mathbf{e} = (0, 0, 0, 0, \alpha^{11}, 0, 0, 0, 0, 0, 0, \alpha^7, 0, 0, 0) \quad (13)$$

Then the errata vector is

$$\bar{\mathbf{u}} = \mathbf{u} + \mathbf{e} = (0, 0, 0, 0, \alpha^{11}, 0, 0, \alpha^2, 0, 0, 0, \alpha^7, 0, 0, 0) \quad (14)$$

Assume the received vector is

$$\mathbf{r} = \mathbf{c} + \bar{\mathbf{u}} = (\alpha^{10}, \alpha^{12}, \alpha^8, \alpha^5, \alpha, \alpha^{14}, \alpha^{13}, \alpha^9, \alpha^9, \alpha^0, \alpha, \alpha^{12}, \alpha^6, \alpha^{12}, \alpha^8) \quad (15)$$

The syndromes S_k for r are

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

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

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

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

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

$$\begin{aligned} T(x) &\equiv \Lambda(x) S(x) \equiv (1 + \alpha^7x)(1 + \alpha^{13}x + \alpha^{14}x^2 + \alpha^{11}x^3 + \alpha x^4 + 0x^5) \bmod x^6 \\ &\equiv (0x^6 + \alpha^8x^5 + \alpha^9x^4 + \alpha x^3 + \alpha^{12}x^2 + \alpha^5x + \alpha^0) \bmod x^6 \\ &= \alpha^8x^5 + \alpha^9x^4 + \alpha x^3 + \alpha^{12}x^2 + \alpha^5x + \alpha^0 \end{aligned} \quad (16)$$

In Eq. (16), 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 polynomial x^{d-1} and $T(x)$ in Eq. (16). By this means, polynomials $\tau(x)$ and $A(x)$ are determined next by use of the Euclidean algorithm. This is accomplished by the recursive Eqs. (9a) and (9b) illustrated in Table 1, where initially $R_{-1}(x) = x^{d-1} = x^6$ and $R_0(x) = \alpha^8x^5 + \alpha^9x^4 + \alpha x^3 + \alpha^{12}x^2 + \alpha^5x + 1$. From Table 1, one observes that $\deg \{R'_s(x)\} = \deg \{R_2(x)\} = 2 \leq \lfloor (d + \nu - 3)/2 \rfloor = 2$. Thus, the computation terminates at $s' = 2$, and

$$R_2(x) = \alpha^7x^2 + \alpha x + \alpha^2 \quad (17a)$$

and

$$\tau_2(x) = \alpha^7x^3 + \alpha^{13}x^2 + \alpha^4x + \alpha^2 \quad (17b)$$

By Eqs. (10a) and (10b), one has

$$\tau(x) = \frac{1}{\alpha^2} \tau_2(x) = \alpha^5 x^3 + \alpha^{11} x^2 + \alpha^2 x + 1 \quad (18)$$

and

$$A(x) = \frac{1}{\alpha^2} R_2(x) = \alpha^5 x^2 + \alpha^{14} x + 1 \quad (19)$$

By using Chien's 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 Eq. (18) is $\tau'(x) = \alpha^5 x^2 + \alpha^2$. Thus, the errata values are

$$Y_1 = \frac{AX_1^{-1}}{\tau'(X_1^{-1})} = \frac{A(\alpha^{-7})}{\tau'(\alpha^{-7})} = \frac{\alpha^5 (\alpha^{-7})^2 + \alpha^{14} (\alpha^{-7}) + 1}{\alpha^5 (\alpha^{-7})^2 + \alpha^2} = \alpha^2$$

$$Y_2 = \frac{AX_2^{-1}}{\tau'(X_2^{-1})} = \frac{\alpha^5 (\alpha^{-3}) + \alpha^{14} (\alpha^{-3}) + 1}{\alpha^5 (\alpha^{-3})^2 + \alpha^2} = \alpha^7$$

and

$$Y_3 = \frac{AX_3^{-1}}{\tau'(X_3^{-1})} = \frac{\alpha^5 (\alpha^{-10})^2 + \alpha^{14} (\alpha^{-10}) + 1}{\alpha^5 (\alpha^{-10})^2 + \alpha^2} = \alpha^{11}$$

III. The Transform Decoder for RS Codes

The transform decoder of Gore and Mandelbaum [17], [18] was developed further in [10] to correct both errors and erasures. This decoding procedure was based on the algorithm originally invented by Forney [13] (also see [10]). By the above-mentioned Euclidean algorithm, the transform domain decoding procedure in [10] can be simplified further.

By the same procedure used in the time domain decoder, one can obtain the errata locator polynomial given in Eq. (4c). Hence,

$$\begin{aligned} \tau(X_i^{-1}) &= 1 + (-1)\tau_1(X_i^{-1}) + (-1)^2\tau_2(X_i^{-1})^2 \\ &+ \dots + (-1)^{\nu+t}\tau_{\nu+t}(X_i^{-1})^{\nu+t} = 0 \\ &\text{for } 1 \leq i \leq \nu + t \end{aligned} \quad (20)$$

Multiplying Eq. (20) by $Y_i X_i^{(b-1)+k}$ yields

$$\begin{aligned} Y_i X_i^{(b-1)+k} - \tau_1 Y_i X_i^{(b-1)+k-1} \\ + \dots + (-1)^{\nu+t}\tau_{\nu+t} Y_i X_i^{(b-1)+k-(\nu+t)} = 0 \end{aligned} \quad (21)$$

Summing Eq. (21) over i for $1 \leq i \leq \nu + t$ produces

$$\begin{aligned} \sum_{i=1}^{\nu+t} Y_i X_i^{(b-1)+k} - \tau_1 \sum_{i=1}^{\nu+t} Y_i X_i^{(b-1)+k-1} \\ + \dots + (-1)^{\nu+t}\tau_{\nu+t} \sum_{i=1}^{\nu+t} Y_i X_i^{(b-1)+k-(\nu+t)} = 0 \end{aligned} \quad (22)$$

From Eq. (22), one has

$$\begin{aligned} S_{(b-1)+k} - \tau_1 S_{(b-1)+k-1} \\ + \dots + (-1)^{\nu+t}\tau_{\nu+t} S_{(b-1)+k-(\nu+t)} = 0 \end{aligned} \quad (23)$$

Hence, in general,

$$\begin{aligned} E_{(b-1)+k} - \tau_1 E_{(b-1)+k-1} \\ + \dots + (-1)^{t+\nu}\tau_{t+\nu} E_{(b-1)+k-(\nu+t)} = 0 \\ \text{for } k \geq d \end{aligned} \quad (24)$$

are the recursive equations for E_j , the transforms of the errata pattern, where initially $E_b = S_b, E_{b+1} = S_{b+1}, \dots, E_{b+d-2}$ are known from the prior syndrome calculation.

From Eq. (24), one obtains the rest of the transform of $\bar{\mathbf{u}}$, i.e., the S_ℓ for $0 \leq \ell \leq N-1$. The amplitude $\bar{\mathbf{u}}$ vector is found by taking the inverse transform over $GF(2^m)$ of $S_\ell, 0 \leq \ell \leq N-1$. Finally, the original m -tuple code vector can be obtained by subtracting $\bar{\mathbf{u}}$ from the received vector \mathbf{r} .

Let us now recapitulate the above transform decoding algorithm of RS codes for correcting both errors and erasures, using transforms over $GF(2^m)$ and the Euclidean algorithm. This procedure is composed of the following five steps:

- (1) Use step 1 in the time domain decoder.
- (2) Use step 2 in the time domain decoder.
- (3) Use step 3 in the time domain decoder.
- (4) Compute the rest of the transform of the errata vector by the use of Eq. (24).
- (5) Invert the transform to recover the errata vector using the fact that $S_0 = S_N$. Then obtain the corrected code vector.

To illustrate the transform domain decoder for correcting errors with erasures, the data for the (15, 9) RS code over $GF(2^4)$ used in Example 1 is again used.

Example 2: Consider the (15, 9) RS code over $GF(2^4)$ with $d=7$. For this code, the erasure, error, errata, and received vectors are given by Eqs. (12), (13), (14), and (15), respectively. By Eq. (18), the errata locator polynomial is

$$\tau(x) = \tau_0 + \tau_1 x + \tau_2 x^2 + \tau_3 x^3 = 1 + \alpha^2 x + \alpha^{11} x^2 + \alpha^5 x^3$$

where $\tau_0 = 1$, $\tau_1 = \alpha^2$, $\tau_2 = \alpha^{11}$, and $\tau_3 = \alpha^5$.

By Eq. (23), the rest of the transform of the errata vector is

$$S_k = \alpha^2 S_{k-1} + \alpha^{11} S_{k-2} + \alpha^5 S_{k-3} \quad \text{for } 7 \leq 15 \quad (25)$$

That is, $S_7 = \alpha^{13}$, $S_8 = \alpha^{13}$, $S_9 = \alpha^7$, $S_{10} = \alpha^3$, $S_{11} = \alpha^5$, $S_{12} = \alpha^{13}$, $S_{13} = \alpha^5$, $S_{14} = \alpha^5$, and $S_{15} = 1$.

The inverse transform of S_k is

$$\bar{u}_k = \sum_{n=0}^{15-1} S_n \alpha^{-nk} \quad \text{for } 0 \leq k \leq 14$$

The result is $\bar{u} = (0, 0, 0, 0, \alpha^{11}, 0, 0, \alpha^2, 0, 0, 0, \alpha^7, 0, 0, 0)$. The corrected code is thus

$$\begin{aligned} \mathbf{c} = \mathbf{r} - \bar{\mathbf{u}} &= (\alpha^{10}, \alpha^{12}, \alpha^8, \alpha^5, \alpha, \alpha^{14}, \alpha^{13}, \alpha^9, \alpha^9, \alpha^0, \alpha, \\ &\quad \alpha^{12}, \alpha^6, \alpha^{12}, \alpha^8) \\ &\quad - (0, 0, 0, 0, \alpha^{11}, 0, 0, \alpha^2, 0, 0, 0, \alpha^7, 0, 0, 0) \\ &= (\alpha^{10}, \alpha^{12}, \alpha^8, \alpha^5, \alpha^6, \alpha^{14}, \alpha^{13}, \alpha^{11}, \alpha^9, \alpha^0, \alpha, \\ &\quad \alpha^2, \alpha^6, \alpha^{12}, \alpha^8) \end{aligned}$$

IV. A Comparison of VLSI Architecture of the Transform Decoder and the Time Domain Decoder

The block diagram of a (255, 223) RS time domain decoder is depicted in Fig. 1. Figure 2 shows the block diagram of a (255, 223) RS transform domain decoder. Each block diagram can be separated into two parts, indicated by broken lines, as shown in both Figs. 1 and 2. The first part, labeled as "I" in both block diagrams, has similar VLSI architecture. The major

functional units in this part are (1) the syndrome computation unit; (2) the power calculation unit; (3) the power expansion unit; (4) the polynomial expansion unit; and (5) the $\lfloor (d + v - 3)/2 \rfloor$ generator. Also included in this part are some delay registers. The lengths of the delay registers may not be equal in these two decoder architectures, but since they contain only replicated register cells, they can be considered identical in architecture.

Figure 3 shows the block diagram of the syndrome computation unit. This unit accepts the received messages and computes their syndromes. There are 32 syndrome subcells in a (255, 223) RS decoder. Each subcell depicted in Fig. 3 performs the operation as $S_i \leftarrow (S_i + r_i \alpha^i)$, where " \leftarrow " denotes the operation "is replaced by." The Berlekamp multiplier is used in this syndrome unit due to its simplicity in VLSI design [19]. The computed syndrome polynomial is labeled as $S(x)$ in both Figs. 1 and 2. In the time domain and transform domain decoders, the coefficients of $S(x)$ are fed in parallel to the polynomial expansion unit 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. Figure 4 shows the block diagram of the power calculation unit. Since the maximum erasure correcting capability of a (255, 223) RS decoder is 32, only 32 symbol latches are needed to store the locations of all the correctable erasures.

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

The power expansion unit converts the α^k 's sequence into an erasure locator polynomial $\Lambda(x)$ which has α^k 's as its roots. Figure 5 depicts the block diagram of this unit. The erasure locator polynomial $\Lambda(x)$ is fed to the modified GCD unit as one of the initial conditions.

A generator is used to compute $\lfloor (d + v - 3)/2 \rfloor$. This is shown in both Figs. 1 and 2. The output of this generator is sent to the modified GCD unit and used as a stop indicator for Euclid's algorithm

Figure 6 presents a block diagram of the polynomial expansion circuit. The Forney syndromes for either the time domain decoder or the transform decoder are calculated in this unit.

Figure 7 depicts the block diagram of the modified GCD unit. As described previously in [6], a multiplexing scheme can be applied to the modified GCD unit to reduce the number of cells needed. The polynomial $\Lambda(x)$ together with the Forney syndrome polynomial $T(x)$ are the two inputs to the modified GCD unit. The output of the modified GCD unit is 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 differing functional units of the time and transform domain decoders are shown in the second half of Figs. 1 and 2 and are labeled as "II." One output of the modified GCD unit of the time domain decoder, the errata locator polynomial $\tau(x)$, 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 $b = 112$ in this design. The other output of the modified GCD unit of the time domain decoder, the errata evaluator polynomial $A(x)$, is fed to the polynomial evaluation unit to perform the evaluation of $A(x)$. Figure 8 shows the block diagram of the polynomial evaluation unit.

The $[x^{111} \tau'(x)]^{-1}$ unit performs the calculation of one part of the errata magnitude [6]. Figure 9 depicts the block diagram of this unit. The product of the outputs of the polynomial evaluation unit and the $[x^{111} \tau'(x)]^{-1}$ unit forms the errata magnitude.

In the time domain decoder, the Chien search unit is used to search for the error and erasure locations; for more details, see [6]. The architecture of the Chien search unit is similar to that of the polynomial evaluation unit, except there is a zero detector at the end.

On the other hand, for the transform domain decoder design, the output from the modified GCD unit is the errata locator polynomial $\tau(x)$. This output is fed to the transform error pattern unit, along with the syndromes from the syndrome computation unit, to calculate the extended syndromes. A new architecture for the transform of the error-pattern unit is developed in Appendix A. The realization of this idea is

shown in the block diagram of the transform of the error-pattern unit, given in Fig. 10.

The computation of extended syndromes, together with the original syndromes, is sent to the inverse transform unit to obtain the estimated error patterns. Figure 11 shows the block diagram of the inverse transform error-pattern unit. It is easy to see that the architecture for the inverse transform unit is similar to that of the syndrome computation unit except that 255 subcells are needed in the inverse transform unit while the syndrome computation unit needs 32 subcells.

Clearly, the architecture of the transform domain decoder design is simpler than that of the time domain decoder design. This is because the transform domain decoder design needs only two regular function blocks in part II of Fig. 2. However, the time domain decoder requires three function blocks for the implementation in part II of Fig. 1.

Furthermore, the inverse-transform unit in the transform domain design contains 255 similar cells in the (255, 223) RS decoder. It is estimated that these 255 cells occupy only a moderate amount of silicon area, and that their geometric arrangement can be regular and simple. Therefore, substantial time for the design and test of such a VLSI chip can be saved. However, the advantage of the transform domain decoder is valid only for moderately short length RS codes. If long length RS codes are used to enhance the system's performance [12], the transform domain decoder needs a large inverse transform block. This might cause a problem in the VLSI implementation. In general, if a $GF(2^m)$ field is used to define an RS code, an inverse transform block composed of $2^m - 1$ cells is needed. Hence, the number of cells needed in an inverse transform block increases exponentially with the integer m . However, the number of transistors needed in the time domain decoder goes up only linearly as the integer m increases. Therefore, for long length codes, the time domain decoder is the more appealing approach. Although the computation of the time domain decoder is more complex than that of the transform domain decoder, for long RS codes the number of transistors needed in a time domain decoder is substantially less than that in a transform domain decoder.

Acknowledgment

We would like to acknowledge the important suggestion of W. L. Eastman of the MITRE Corp. that both the errata evaluator and locator polynomials can be obtained directly by only a modification of the initial conditions of the Berlekamp-Massey algorithm or Euclid's algorithm.

References

- [1] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes," *IEEE Trans. on Contr.*, vol. 27, pp. 87-99, 1975.
- [2] I. S. Reed, R. A. Scholtz, T. K. Truong, and L. R. Welch, "The Fast Decoding of Reed-Solomon Codes Using Fermat Theoretic Transforms and Continued Fractions," *IEEE Trans. on Information Theory*, vol. IT-24, no. 1, pp. 100-106, January 1978.
- [3] L. R. Welch and R. A. Scholtz, "Continued Fractions and Berlekamp's Algorithm," *IEEE Trans. on Information Theory*, vol. IT-25, pp. 19-27, 1979.
- [4] R. P. Brent and H. T. Kung, "Systolic VLSI Arrays for Polynomial GCD Computation," *IEEE Trans. on Computers*, vol. C-33, no. 8, pp. 731-736, August 1984.
- [5] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen, and I. S. Reed, "A VLSI Design of a Pipeline Reed-Solomon Decoder," *IEEE Trans. on Computers*, vol. C-34, no. 5, pp. 393-403, May 1985.
- [6] H. M. Shao, T. K. Truong, I. S. Hsu, L. J. Deutsch, and I. S. Reed, "A Single Chip VLSI Reed-Solomon Decoder," *TDA Progress Report 42-84*, vol. October-December 1985, Jet Propulsion Laboratory, Pasadena, California, pp. 73-81, February 15, 1986.
- [7] R. E. Blahut, *Theory and Practice of Error Control Codes*, New York: Addison-Wesley, p. 258, May 1984.
- [8] W. L. Eastman, *Decoding Erasures*, Bedford, Massachusetts: Mitre Corporation, 1986.
- [9] T. K. Truong, W. L. Eastman, I. S. Reed, and I. S. Hsu, "A Simplified Procedure for Correcting Both Errors and Erasures of a Reed-Solomon Code Using the Euclidean Algorithm," *TDA Progress Report 42-91*, vol. July-September 1987, Jet Propulsion Laboratory, Pasadena, California, pp. 200-212, November 15, 1987.
- [10] I. S. Reed, T. K. Truong, and R. L. Miller, "Simplified Algorithm for Correcting Both Errors and Erasures of Reed-Solomon Codes," *Proc. IEE*, vol. 126, no. 10, pp. 961-963, October 1979.
- [11] R. L. Miller, L. J. Deutsch, and S. A. Butman, *On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes*, JPL Publication 81-9, Jet Propulsion Laboratory, Pasadena, California, September 1981.
- [12] J. H. Yuen and Q. D. Vo, "In Search of a 2-dB Coding Gain," *TDA Progress Report 42-83*, vol. July-September 1985, Jet Propulsion Laboratory, Pasadena, California, pp. 26-33, November 15, 1985.
- [13] G. D. Forney, "On Decoding BCH Codes," *IEEE Trans. on Information Theory*, vol. IT-11, pp. 549-557, 1965.
- [14] E. R. Berlekamp, *Algebraic Coding Theory*, New York: McGraw-Hill, 1968.
- [15] I. S. Reed, T. K. Truong, and R. L. Miller, "Decoding of BCH and RS Codes With Errors and Erasures Using Continued Fractions," *Electronics Letters*, vol. 15, no. 17, pp. 542-544, August 16, 1976.
- [16] E. R. Berlekamp, "Bit-Serial Reed-Solomon Encoders," *IEEE Trans. on Information Theory*, vol. IT-28, no. 6, pp. 869-874, November 1982.

- [17] W. C. Gore, *Transmitting Binary Symbols With Reed–Solomon Code*, Johns Hopkins Electrical Engineering Department Report 73-75, April 1973.
- [18] D. Mandelbaum, "On Decoding Reed–Solomon Codes," *IEEE Trans. on Information Theory*, vol. IT-17, pp. 707–712, 1971.
- [19] I. S. Hsu, T. K. Truong, H. M. Shao, L. J. Deutsch, and I. S. Reed, "A Comparison of VLSI Architecture of Finite Field Multipliers Using Dual, Normal or Standard Bases," *TDA Progress Report 42-90*, vol. April–June 1987, Jet Propulsion Laboratory, Pasadena, California, pp. 63–75, August 15, 1987.
- [20] L. Johnson, U. Weiser, D. Cohen, and A. Davis, "Towards a Formal Treatment of VLSI Arrays," presented at the Caltech Conference on VLSI, Pasadena, California, January 1981.

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			$\alpha^8 x^5 + \alpha^9 x^4 + \alpha x^3 + \alpha^{12} x^2 + \alpha^5 x + 1$	$1 + \alpha^7 x$
1	$\frac{\alpha^8 x^5 + \alpha^9 x^4 + \alpha x^3 + \alpha^{12} x^2 + \alpha^5 x + 1}{x^6 + \alpha x^5 + \alpha^8 x^4 + \alpha^4 x^3 + \alpha^{12} x^2 + \alpha^7 x}$ $\frac{\frac{1}{\alpha^8} x + \frac{\alpha}{\alpha^8}}{\alpha x^5 + \alpha^2 x^4 + \alpha^9 x^3 + \alpha^5 x^2 + \alpha^{13} x + \alpha^8}$ $\frac{\alpha x^4 + \alpha^{14} x^3 + \alpha^{14} x^3 + \alpha^3 x + \alpha^8}{\alpha x^4 + \alpha^{14} x^3 + \alpha^{14} x^3 + \alpha^3 x + \alpha^8}$	$\frac{1}{\alpha^8}(x + \alpha)$	$x^4 + \alpha^{14} x^3 + \alpha^6 x^2 + \alpha^2 x + \alpha^8$	$\frac{1}{\alpha^8}(x + \alpha)(1 + \alpha^7 x)$ $+ 0 = \frac{1}{\alpha^8}(\alpha^7 x^2 + \alpha^2 x + 2)$
2	$\frac{\alpha^8 x + 1}{\alpha^8 x^5 + \alpha^9 x^4 + \alpha x^3 + \alpha^{12} x^2 + \alpha^5 x + \alpha^8}$ $\frac{\alpha^8 x^4 + \alpha^{14} x^3 + \alpha^{14} x^3 + \alpha^3 x + \alpha^8}{\alpha^8 x^5 + \alpha^9 x^4 + \alpha^7 x^3 + \alpha^3 x^2 + \alpha x}$ $\frac{x^4 + \alpha^{14} x^3 + \alpha x^2 + \alpha^2 x + 1}{x^4 + \alpha^{14} x^3 + \alpha^{14} x^3 + \alpha^3 x + \alpha^8}$ $\frac{\alpha^7 x^2 + \alpha x + \alpha^2}{\alpha^7 x^2 + \alpha x + \alpha^2}$	$\alpha^8 x + 1$	$\alpha^7 x^2 + \alpha x + \alpha^2$	$(\alpha^7 x^2 + \alpha^2 x + \alpha) \cdot (x + \alpha^7)$ $+ (1 + \alpha^7 x)$ $= \alpha^7 x^3 + \alpha^{13} x^2 + \alpha^4 x + \alpha^2$

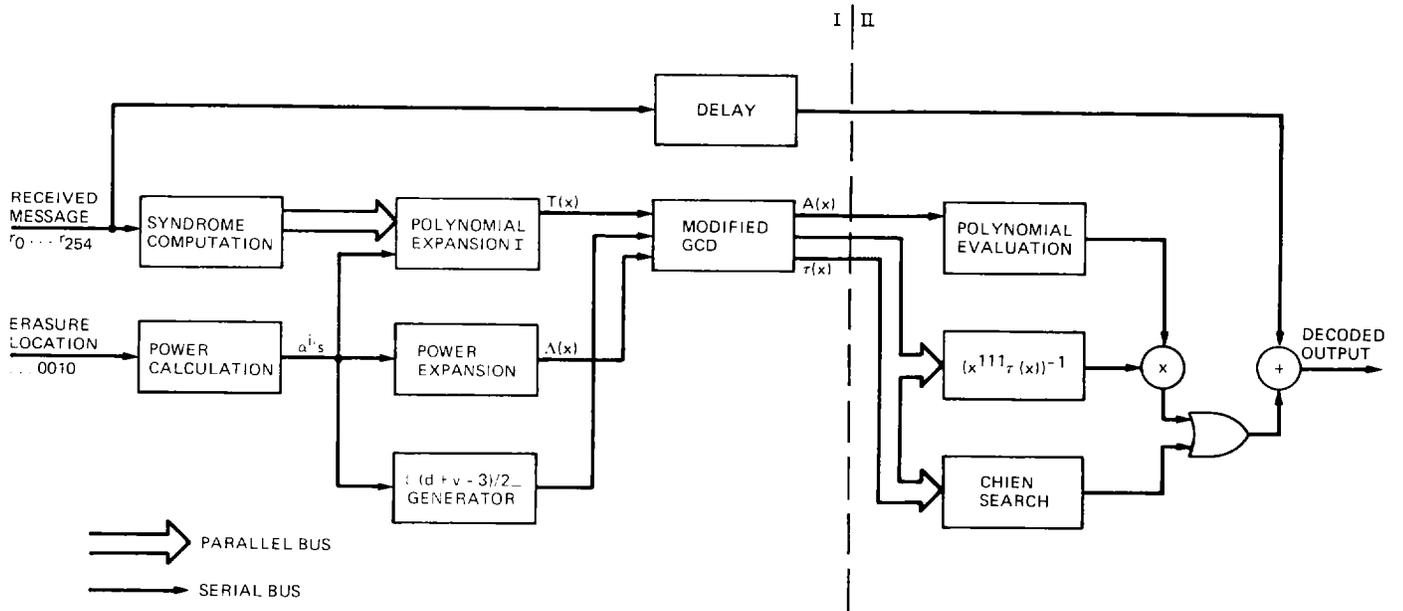


Fig. 1. Overall block diagram of a pipeline (255,223) RS time domain decoder

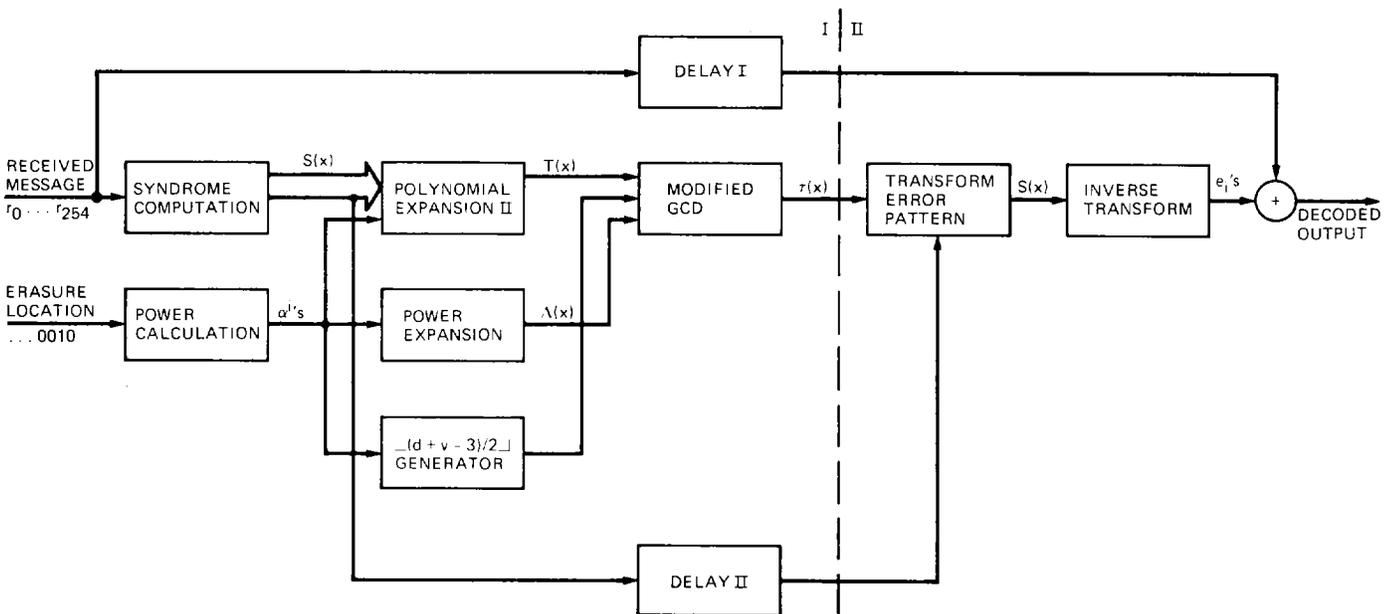


Fig. 2. Overall block diagram of a pipeline (255,223) RS transform domain decoder

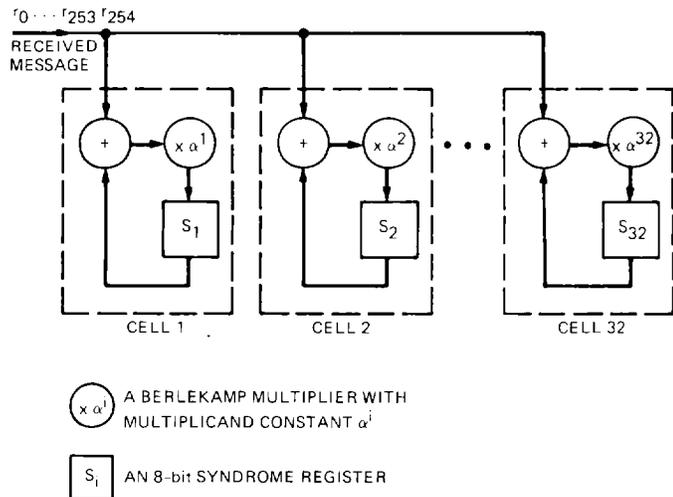


Fig. 3. Block diagram of the syndrome computing chip

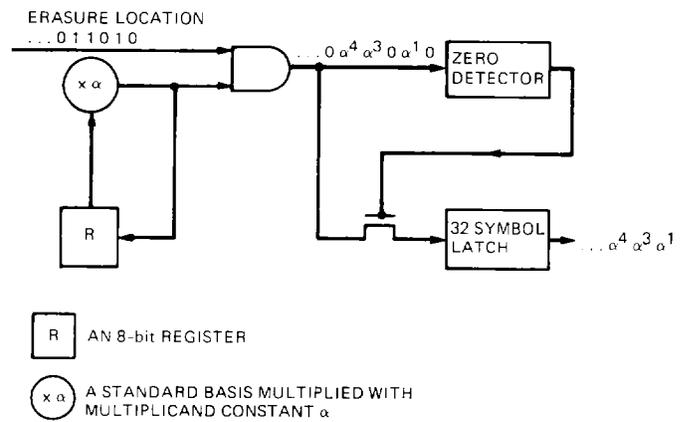


Fig. 4. Block diagram of the power calculation unit

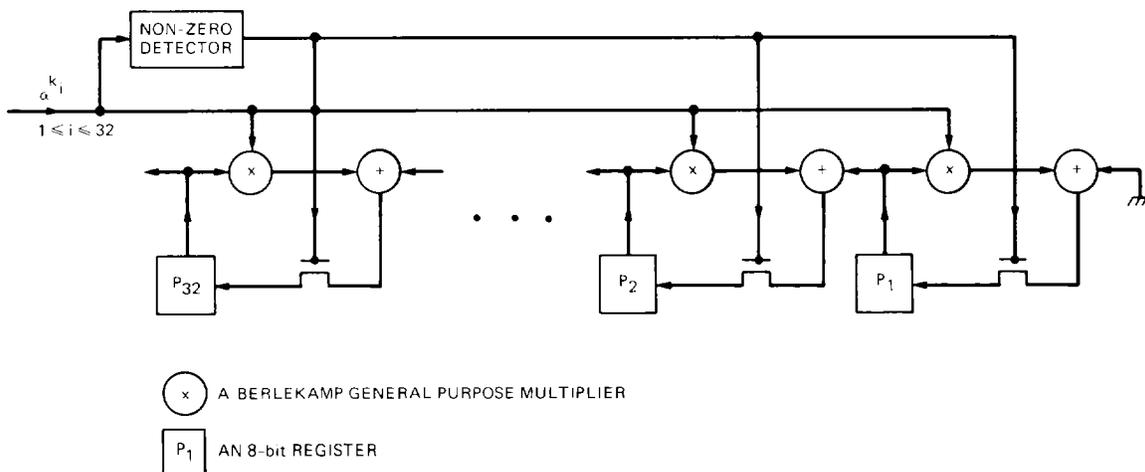


Fig. 5. Block diagram of the power expansion unit

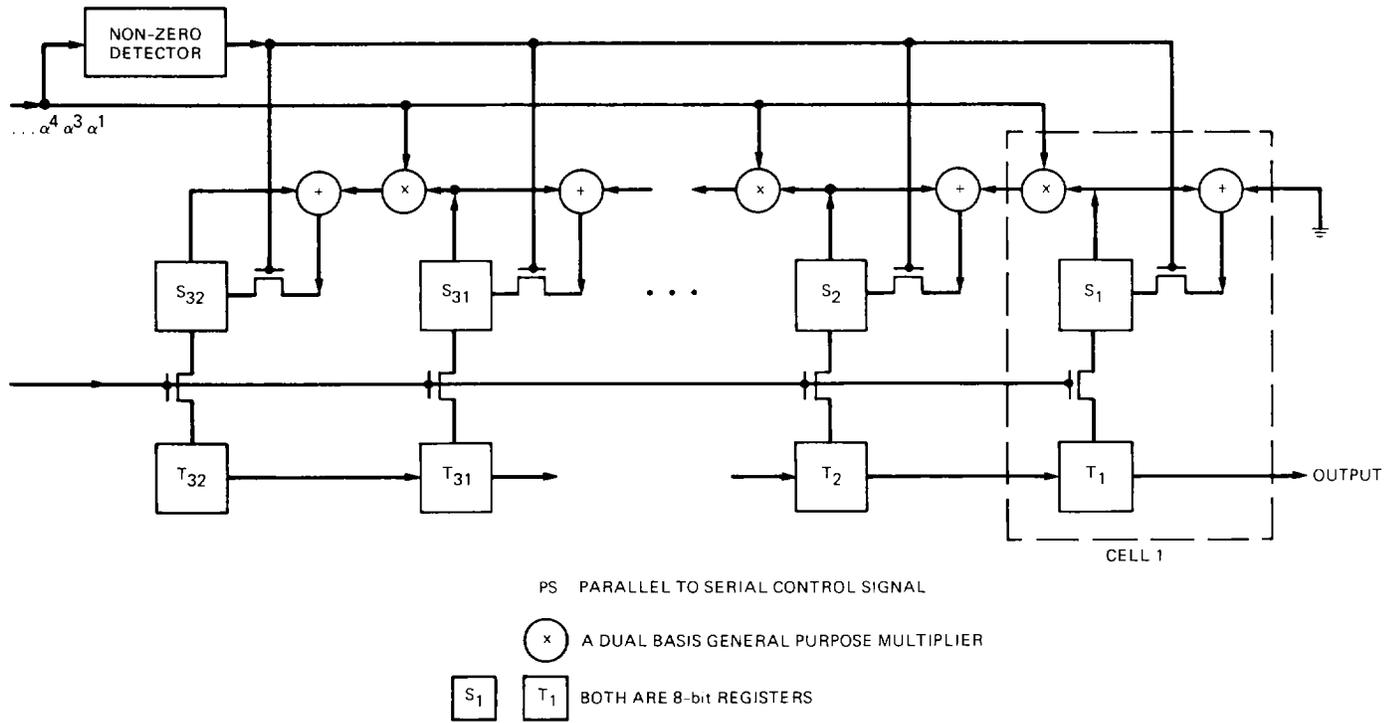


Fig. 6. Block diagram of a polynomial expansion unit

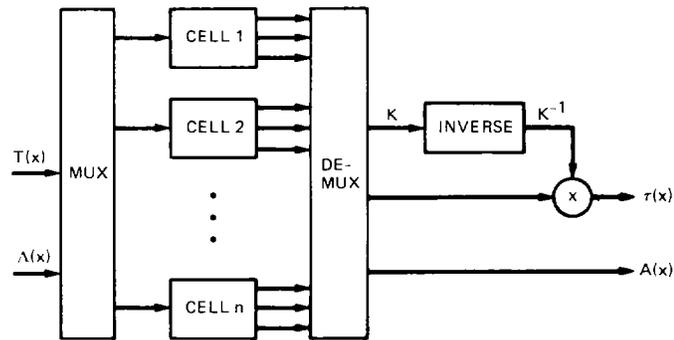
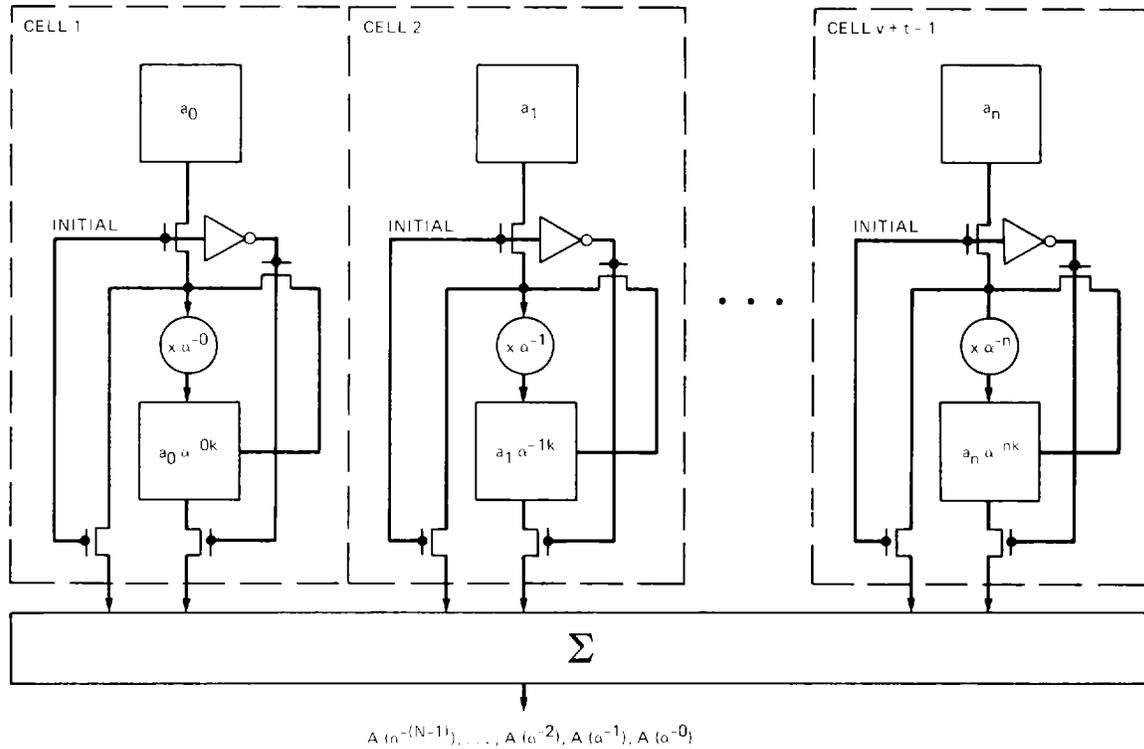


Fig. 7. Block diagram of the modified GCD unit



 A BERLEKAMP MULTIPLIER WITH MULTIPLICAND FIXED AS α^i
 AN 8-bit REGISTER
 $n = v + t - 1$

Fig. 8. Block diagram of the polynomial evaluation unit

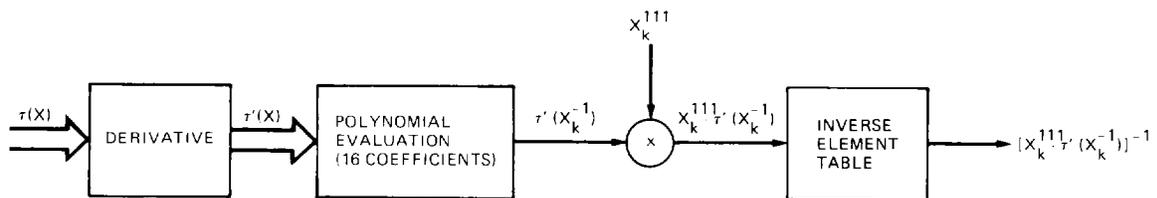


Fig. 9. Block diagram of the $[x^{111} r'(x)]^{-1}$ unit

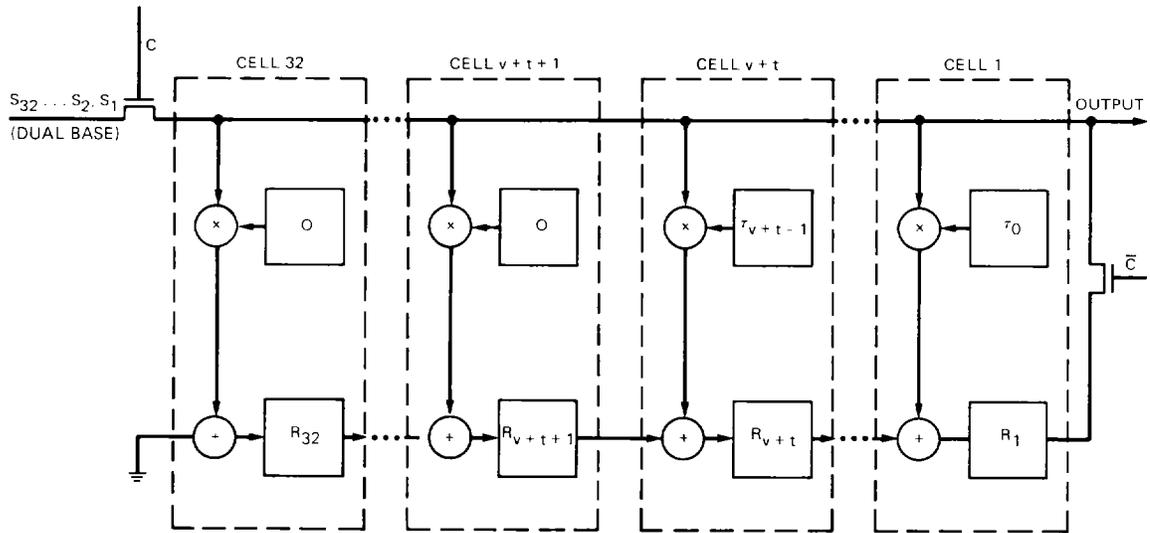


Fig. 10. Block diagram of the transform of the error pattern

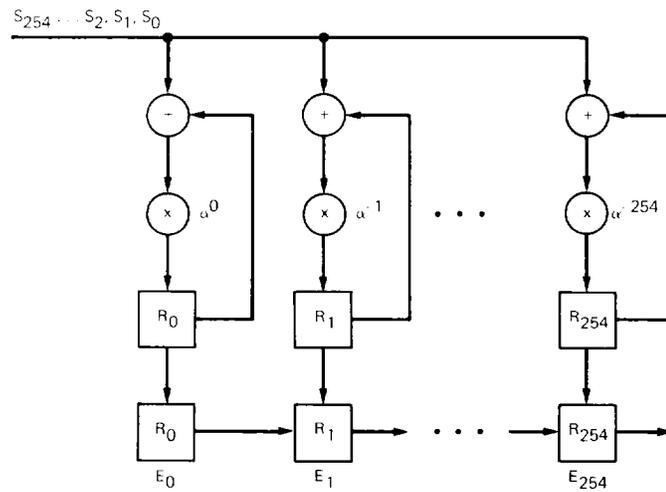


Fig. 11. Block diagram of the inverse transform unit

Appendix A

New Architecture for the Transform of the Error Pattern Unit

In this appendix, a VLSI architecture is developed to compute the transform of the error pattern. Recently, Johnson *et al.* [20] proposed a systolic array for computing a linear convolution. Using a technique similar to that suggested by Johnson *et al.*, the transform of the error pattern in Eq. (23) or Eq. (24) can be implemented in a systolic array. The advantage of this method over the previous method in Fig. 15 of [5] is that the long delay needed in the large XOR tree used for summing all the terms in Eq. (23) is eliminated. Also, the zero detectors needed in the previous design [5] are not required in this new architecture.

To illustrate this new architecture, the data in Example 2 for a (15, 9) RS code are used here as an example. The recursive equation to compute the remainder of the transform of the error pattern is given in Eq. (25). The new design for computing Eq. (25) is shown in Fig. A-1. In this figure, the function of each cell can be described by a register transfer

relation of the type $R_i \leftarrow R_{i+1} + S_k \alpha^i$. The input data are sent to all the cells simultaneously.

To understand the operation of this circuit, assume initially that all registers R_i for $1 \leq i \leq 3$ are set to zero. The control signal C is high for 6 symbol clocks to allow data S_1, S_2, \dots, S_6 to be fed into the circuit. The input data are also sent to the output node. At the same time, the complement signal \bar{C} of signal C is low to prevent the data stored in register R_1 from being sent to the output node. Note that one "clock time" for one Galois field symbol equals 4 circuit clock times. At the seventh symbol clock time the control signal C is switched to low or zero so that $\bar{C} = 1$. Therefore, the data stored in register R_1 , which equals S_7 at that moment, is sent to the output node and fed back to all basic cells. This process continues until the rest of the transform of the error pattern, i.e., S_7, S_8, \dots, S_{15} , is obtained. The detailed operation of this circuit is described in Table A-2.

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

Table A-2. The fifteen steps of the transform of the error pattern algorithm

Symbol clock time	R_1	R_2	R_3
1	$\alpha^5 S_1$	$\alpha^{11} S_1$	$\alpha^2 S_1$
2	$\alpha^5 S_2$	$\alpha^5 S_1 + \alpha^{11} S_2$	$\alpha^{11} S_1 + \alpha^2 S_2$
3	$\alpha^5 S_3$	$\alpha^5 S_2 + \alpha^{11} S_3$	$\alpha^2 S_3 + \alpha^{11} S_2 + \alpha^5 S_1 = S_4$
4	$\alpha^5 S_4$	$\alpha^{11} S_4 + \alpha^5 S_3$	$\alpha^2 S_4 + \alpha^{11} S_3 + \alpha^5 S_2 = S_5$
·	·	·	·
·	·	·	·
·	·	·	·
15	$\alpha^5 S_{14}$	$\alpha^{11} S_{14} + \alpha^5 S_{13}$	$\alpha^2 S_{14} + \alpha^{11} S_{13} + \alpha^5 S_{12} = S_{15} = S_0$

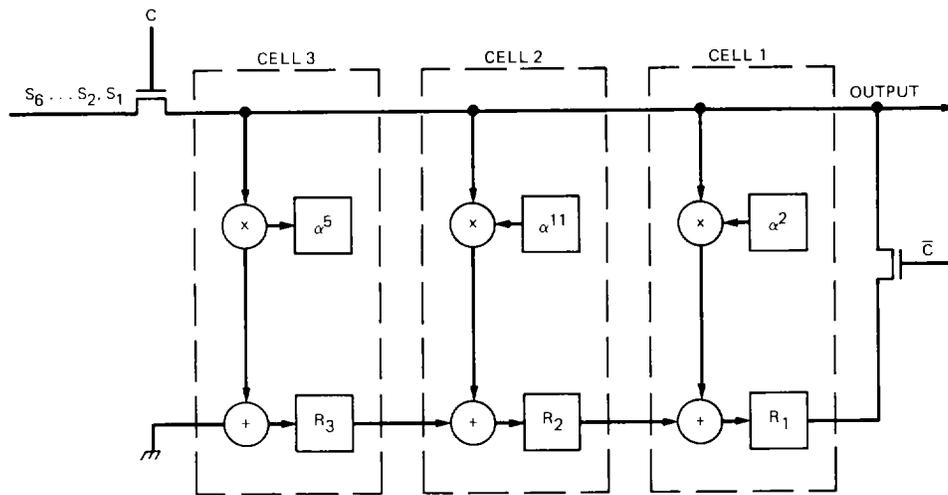


Fig. A-1. A systolic array for computing the transform of the error pattern in Eq. (25)