On the VLSI Design of a Pipeline Reed–Solomon Decoder Using Systolic Arrays

H. M. Shao and L. J. Deutsch
Communications Systems Research Section
I. S. Reed
University of Southern California

A new VLSI design of a pipeline Reed-Solomon decoder is presented. The transform decoding technique used in a previous article is replaced by a time domain algorithm through a detailed comparison of their VLSI implementations. A new architecture that implements the time domain algorithm permits efficient pipeline processing with reduced circuitry. Erasure correction capability is also incorporated with little additional complexity. By using a multiplexing technique, a new implementation of Euclid's algorithm maintains the throughput rate with less circuitry. Such improvements result in both enhanced capability and significant reduction in silicon area.

I. Introduction

Recently a VLSI design of a pipeline Reed-Solomon decoder was presented [1]. A modified form of Euclid's algorithm was developed which avoided computations of inverse elements. A systolic array architecture was designed, from a suggestion by Brent and Kung [2], to implement the modified Euclid's algorithm. More recently, another VLSI design of an RS decoder was introduced [3]. It combined the algorithm in [4] and the modified Euclid's algorithm instead of the continued fraction technique. The decoder design in [3] used a time domain decoding algorithm to reduce the massive circuitry required by the inverse transform in [1]. The decoder design also included the erasure correction capability, and, during the design process, a recursive architecture was derived to implement the modified Euclid's algorithm by far fewer circuits than used in [1].

It has been pointed out [5] that the errata locator polynomial can be obtained directly from the Massey-Berlekemp algorithm if initialized properly. This suggestion led to improvements in the VLSI design in [3].

In this article, an efficient time domain RS decoding algorithm is described and verified. It is shown that the modified Euclid's algorithm can produce the errata locator polynomial and errata evaluator polynomial simultaneously, similar to the Massey-Berlekemp algorithm. The VLSI architectures for syndrome computations, polynomial expansions, modified Euclid's algorithm performance, and polynomial evaluations are also described.

This work was carried out during the architectural phase of the Advanced Reed-Solomon Decoder (ARSD) project and

should be viewed as a companion to the recent work of Truong, *et al.* [6]. In that article, a transform domain decoder architecture is developed which, due to its design simplicity, has been chosen for the prototype VLSI implementation of the ARSD. However, the work presented here and in [6] clearly shows that the time domain architecture has many desirable features which make it an attractive candidate for future VLSI implementation.

II. The Time Domain Reed–Solomon Decoding Algorithm

Let $N = 2^m - 1$ be the length of the (N, I) RS code with design distance d.

Let

$$R(X) = \sum_{i=0}^{N-1} r_i X^i = r_{N-1} X^{N-1} + \ldots + r_1 X + r_0$$

be the received message. Suppose e errors and E erasures occur, and $2e + E < d - 1$. Define $\Lambda = \{\alpha^{-i} | r_i$ declared as an erasure$\}$.

The decoding algorithm is as follows:

Step 1. Compute the syndromes

$$S_k = \sum_{i=0}^{N-1} r_i X^i \Big|_{X=\alpha^k}$$

$$= \sum_{i=0}^{N-1} r_i \alpha^{ki} \qquad \text{for } 1 \leqslant k \leqslant d-1 \tag{1}$$

Form a syndrome polynomial

$$S(X) = \sum_{k=1}^{d-1} S_k X^{k-1} \tag{2}$$

Step 2. Compute the erasure locator polynomial $\Lambda(X)$. Assume the erasure location information is received in the form of a binary sequence synchronous to the received message

$$R(X) = \sum_{i=0}^{N-1} r_i X^i$$

Then for each symbol r_i that is labeled as an erasure, α^{-i} should be the root of the erasure locator polynomial $\Lambda(X)$. That is,

$$\Lambda(X) = \prod_{\alpha^{-i} \in \Lambda} (X - \alpha^{-i}) \tag{3}$$

Step 3. Multiply the syndrome polynomial $S(X)$ by the erasure locator polynomial $\Lambda(X)$ to form the modified syndrome polynomial

$$T(X) = S(X)\,\Lambda(X) \bmod X^{d-1}$$

$$= \sum_{k=1}^{d-1} T_k X^{k-1} \tag{4}$$

Step 4. If $\deg(\Lambda(X)) > \deg(T(X))$, then no error has occurred, i.e., $e = 0$. Thus there is no need to perform the modified Euclid's algorithm. Let the errata locator polynomial $\sigma(X) = \Lambda(X)$ and the errata evaluator polynomial $\omega(X) = T(X)$. If $\deg(\Lambda(X)) \leqslant \deg(T(X))$, then perform a modified Euclid's algorithm on X^{d-1} and $T(X)$ with the following initializations:

$$\begin{aligned} \mu_0(X) &= \Lambda(X) & R_0(X) &= X^{d-1} \\ \lambda_0(X) &= 0 & Q_0(X) &= T(X) \end{aligned} \tag{5}$$

Compute the following iterations:

$$\begin{aligned} R_i(X) = {} & [\sigma_{i-1} b_{i-1} R_{i-1}(X) + \bar{\sigma}_{i-1} a_{i-1} Q_{i-1}(X)] \\ & - X^{|\varrho_{i-1}|} [\sigma_{i-1} a_{i-1} Q_{i-1}(X) \\ & + \bar{\sigma}_{i-1} b_{i-1} R_{i-1}(X)] \end{aligned} \tag{6}$$

$$\begin{aligned} \lambda_i(X) = {} & [\sigma_{i-1} b_{i-1} \lambda_{i-1}(X) + \bar{\sigma}_{i-1} a_{i-1} \mu_{i-1}(X)] \\ & - X^{|\varrho_{i-1}|} [\sigma_{i-1} a_{i-1} \mu_{i-1}(X) \\ & + \bar{\sigma}_{i-1} b_{i-1} \lambda_{i-1}(X)] \end{aligned} \tag{7}$$

$$Q_i(X) = \sigma_{i-1} Q_{i-1}(X) + \bar{\sigma}_{i-1} R_{i-1}(X) \tag{8}$$

$$\mu_i(X) = \sigma_{i-1} \mu_{i-1}(X) + \bar{\sigma}_{i-1} \lambda_{i-1}(X) \tag{9}$$

where a_{i-1} and b_{i-1} are the leading coefficients of $R_{i-1}(X)$ and $Q_{i-1}(X)$, respectively,

$$\ell_{i-1} = \deg(R_{i-1}(X)) - \deg(Q_{i-1}(X))$$

and

$$\begin{aligned} \sigma_{i-1} &= 1 \qquad \text{if } \ell_{i-1} \geqslant 0 \\ \sigma_{i-1} &= 0 \qquad \text{if } \ell_{i-1} < 0 \end{aligned} \tag{10}$$

Stop the iterations when $\deg(\lambda_i(X)) > \deg(R_i(X))$. Let the errata locator polynomial $\sigma(X) = \lambda_i(X)$ and the errata evaluator polynomial $\omega(X) = R_i(X)$. The $\sigma(X)$ and $\omega(X)$ polynomials, obtained by the modified Euclid's algorithm, both carry a common scale factor compared to those computed by the conventional Euclid's algorithm. But this scale factor does not affect the errata location computations or the errata magnitude computations.

Step 5. Evaluate the errata locator polynomial $\sigma(X)$ for α^{-i}, $i = 0, \ldots, N-1$ to find the roots of $\sigma(X)$. If $\sigma(\alpha^{-i}) = 0$, then r_i is a corrupted symbol.

Step 6. Compute the corresponding errata magnitudes by evaluating $\omega(X)$ and $\sigma'(X)$ for α^{-i}, $i = 0, \ldots, N-1$. That is, the errata magnitude

$$\hat{e}_i = -\frac{\omega(\alpha^{-i})}{\sigma'(\alpha^{-i})} \qquad 0 \leqslant i \leqslant N-1 \tag{11}$$

Note that the scale factor carried by $\omega(X)$ and $\sigma(X)$ is automatically cancelled by this division.

Step 7. Subtracting $\hat{e}_i$ from r_i yields the decoded codeword

$$\hat{C}_i = r_i - \hat{e}_i \qquad 0 \leqslant i \leqslant N-1 \tag{12}$$

Note that the modified Euclid's algorithm in Step 4 is a combination of three techniques. First, observe that the error locator polynomial $\lambda(X)$ and the errata evaluator polynomial $\omega(X)$ can be obtained from Euclid's algorithm by computing the GCD of the modified syndrome $T(X)$ and X^{d-1} with the following initializations:

$$\begin{aligned} \mu_0(X) &= 1 \qquad & R_0(X) &= X^{d-1} \\ \lambda_0(X) &= 0 \qquad & Q_0(X) &= T(X) \end{aligned} \tag{13}$$

Since e errors and E erasures occur and $2e + E \leqslant d-1$, as in Theorem 8.4 of [7], the following properties hold:

$$\deg(\lambda(X)) = e \qquad \deg(\omega(X)) < e + E \tag{14}$$

$$\text{GCD}(\lambda(X), \omega(X)) = 1 \tag{15}$$

$$e_i = -\frac{\omega(X)}{[\Lambda(X)\lambda(X)]'}\bigg|_{X=\alpha^{-i}} \tag{16}$$

$$\lambda(X)\,\Lambda(X)S(X) \equiv \omega(X) \bmod X^{d-1} \tag{17}$$

Applying properties (14) and (17) to Theorem 8.5 of [7] implies that there exist a unique j and a unique polynomial $\beta(X)$ such that

$$\lambda(X) = \beta(X)\,\lambda_j(X)$$

$$\omega(X) = \beta(X)\,R_j(X)$$

By properties (15) and (16), $\beta(X)$ is a constant, which can be taken to be unity without affecting the roots of $\lambda(X)$ or the magnitudes e_i. The second technique applied to the modified Euclid's algorithm is that the errata locator polynomial $\sigma(X) = \Lambda(X)\,\lambda(X)$ can be obtained directly from the Euclid's algorithm. To achieve this, $\mu_o(X)$ must be initialized to be the erasure locator polynomial $\Lambda(X)$ instead of 1, and the iteration stop criterion must be changed to $\deg(R_i(X)) < \deg(\lambda_i(X))$. Such a change simply results in all $\lambda_i(X)$ carrying the factor $\Lambda(X)$. The errata evaluator polynomial $\omega(X)$ is not affected by such initialization because $\lambda_i(X)$ does not involve the computation of $R_i(X)$. As will be shown later, using the modified Euclid's algorithm to compute the errata locator polynomial directly eliminates the need for polynomial multiplication circuits and delay lines in a VLSI pipeline implementation. Thirdly, the modified Euclid's algorithm uses cross multiplication and subtraction to replace polynomial division. Such operations eliminate the need to compute finite field inverse elements, which is performed by a table look-up, in this step. Since a look-up table involves the use of a large silicon area in VLSI, it is preferable to do this as infrequently as possible.

Example. Consider an RS (8, 4) code over GF (17) with generator polynomial $g(X) = (X-2)(X-2^2)(X-2^3)(X-2^4)$. Suppose two erasures and one error have occurred and the all zero codeword was sent. Let $R(X) = -2X^5 - 3X^2 + 2X$ be the received vector with locations X^5 and X^2 flagged as erasures. Thus the erasure locator polynomial

$$\Lambda(X) = (X - 2^{-5})(X - 2^{-2})$$

$$= X^2 + 13X + 2$$

(1) Compute the syndromes

$$S_k = \sum_{i=0}^{7} r_i 2^{ik} \qquad k = 1, 2, 3, 4$$

$$S_1 = R(2^1) = 13$$

$$S_2 = R(2^2) = 3$$

$$S_3 = R(2^3) = 10$$

$$S_4 = R(2^4) = 14$$

Form the syndrome polynomial

$$S(X) = \sum_{k=1}^{4} S_k X^{k-1} = S_4 X^3 + S_3 X^2 + S_2 X^1 + S_1$$

$$= 14X^3 + 10X^2 + 3X + 13$$

(2) Compute the modified syndromes

$$T(X) = S(X)\,\Lambda(X) \bmod X^4$$

$$= (14X^3 + 10X^2 + 3X + 13) \times (X^2 + 13X + 2) \bmod X^4$$

$$= 8X^3 + 4X^2 + 5X + 9$$

Thus

$$T_4 = 8, T_3 = 4, T_2 = 5, T_1 = 9$$

(3) Perform the modified Euclid's algorithm

$$\mu_0(X) = \Lambda(X) = X^2 + 13X + 2$$

$$\lambda_0(X) = 0$$

$$R_0(X) = X^4$$

$$Q_0(X) = T(X) = 8X^3 + 4X^2 + 5X + 9$$

$$R_1(X) = 8R_0(X) - XQ_0(X)$$

$$= 8X^4 - X(8X^3 + 4X^2 + 5X + 9)$$

$$= -4X^3 - 5X^2 - 9X$$

$$\lambda_1(X) = 8\lambda_0(X) - X\mu_0(X) = -X(X^2 + 13X + 2)$$

$$= -X^3 - 13X^2 - 2X$$

$$Q_1(X) = Q_0(X) = 8X^3 + 4X^2 + 5X + 9$$

$$\mu_1(X) = \mu(X) = X^2 + 13X + 2$$

$$R_2(X) = 8R_1(X) - (-4)Q_1(X)$$

$$= 8(-4X^3 - 5X^2 - 9X) + 4(8X^3 + 4X^2 + 5X + 9)$$

$$= 10X^2 - X + 2$$

$$\lambda_2(X) = 8\lambda_1(X) - (-4)\mu_1(X) = 8(X^3 - 13X^2 - 2X) + 4(X^2 + 13X + 2)$$

$$= 9X^3 + 2X^2 + 2X + 8$$

Since $\deg(\lambda_2(X)) - \deg(R_2(X)) = 1$, Stop.

Thus the errata evaluator is

$$\omega(X) = R_2(X) = 10X^2 - X + 2$$

and the errata locator is

$$\sigma(X) = \lambda_2(X) = 9X^3 + 2X^2 + 2X + 8$$

(4) Perform Chien search on $\sigma(X)$ and evaluate $-\omega(X)/\sigma'(X)$

$\sigma(2^{-7}) = 7;$ $\quad \hat{e}_7 = 0$

$\sigma(2^{-6}) = 12;$ $\quad \hat{e}_6 = 0$

$\sigma(2^{-5}) = 0;$ $\quad \hat{e}_5 = -\dfrac{\omega(2^{-5})}{\sigma'(2^{-5})} = -2$

$\sigma(2^{-4}) = 16;$ $\quad \hat{e}_4 = 0$

$\sigma(2^{-3}) = 8;$ $\quad \hat{e}_3 = 0$

$\sigma(2^{-2}) = 0;$ $\quad \hat{e}_2 = -\dfrac{\omega(2^{-2})}{\sigma'(2^{-2})} = -3$

$\sigma(2^{-1}) = 0;$ $\quad \hat{e}_1 = -\dfrac{\omega(2^{-1})}{\sigma'(2^{-1})} = 2$

$\sigma(2^{-0}) = 4;$ $\quad \hat{e}_0 = 0$

(5) $C_i = r_i - \hat{e}_i \qquad i = 7, 6, 5, 4, 3, 2, 1, 0$

$= (0, 0, -2, 0, 0, -3, 2, 0) - (0, 0, -2, 0, 0, -3, 2, 0)$

$= (0, 0, 0, 0, 0, 0, 0, 0)$

The VLSI architecture of the pipeline RS decoder is shown in Fig. 1. The syndromes $S(X)$ are computed by a form of polynomial evaluation. The α^k generation block converts binary erasure location information to powers of α which are the roots of the erasure locator polynomial. The modified syndromes $T(X)$ and the erasure locator polynomial $\Lambda(X)$ can be computed by two polynomial multiplication circuits. By the use of a multiplexing and recursive technique, the modified Euclid's algorithm is implemented with a significant reduction of cells over a previous design [1]. The errata evaluator polynomial $\omega(X)$ and the errata locator polynomial $\sigma(X)$ are then evaluated using two polynomial evaluation circuits different from the one used for syndrome computation. The errata locations thus obtained direct the subtractions of the errata from the received messages to produce the decoded messages. In the following, the VLSI design of each functional block is described.

III. VLSI Implementation of the Syndrome Computation

The syndrome computation

$$S_k = \sum_{i=0}^{N-1} r_i \alpha^{ki} \qquad 1 \leqslant k \leqslant d-1 \qquad (18)$$

is an evaluation of a polynomial of length N on $d-1$ points. Since $N > d-1$, it is best to compute all syndromes simultaneously in the following manner as each r_i is received:

$$S_k = \left(\ldots (r_{N-1} \alpha^k + r_{N-2}) \alpha^k + \ldots + r_1 \right) \alpha^k + r_0 \qquad (19)$$

Note that r_{N-1} is the first received symbol. Starting from the innermost parentheses, syndrome S_k is gradually computed as r_i are received. After r_0 is entered, all $d-1$ syndrome computations are completed at the same time. They are ready to be shifted out serially at that point. A systolic array design of a syndrome computation circuit is shown in Fig. 2.

IV. A VLSI Design for Polynomial Expansion

Recall that Λ is the set of α^{-i} where $\alpha^{-i} \epsilon \Lambda$ implies the location of r_i is an erasure. The computation of the erasure locator polynomial $\Lambda(X)$ demands the expansion of

$$\Lambda(X) = \prod_{\alpha^{-i} \epsilon \Lambda} (X - \alpha^{-i}) \qquad (20)$$

from one root α^{-i} at a time. Similarly, the modified syndromes

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

$$\equiv S(X) \prod_{\alpha^{-i} \epsilon \Lambda} (X - \alpha^{-i}) \bmod X^{d-1} \qquad (21)$$

can also be computed in the same manner except $T(X)$ uses $S(X)$, instead of 1, as an initial condition. Therefore, a polynomial expansion circuit is developed to calculate $T(X)$ and $\Lambda(X)$.

Note that for an arbitrary $S(X)$, which may be 1,

$$S(X)(X - \alpha^{-i}) = XS(X) - \alpha^{-i} S(X) \qquad (22)$$

This computation can be accomplished by a linear shift of $S(X)$, multiplication of every coefficient of $S(X)$ by α^{-i}, and finite field additions. A systolic array is designed, as shown in Fig. 3, to implement such simple operations. The control signal "zero" ensures that the resultant polynomial would not be changed if $\alpha^{-i} = 0$.

V. A New Architecture to Perform the Modified Euclidean Algorithm

A systolic array was designed in [2] to compute the error locator polynomial by a modified Euclidean algorithm. The array required $2t$ cells, twice the number of correctable errors. It is capable of performing the modified Euclidean algorithm continuously.

In the modified Euclidean algorithm only one syndrome polynomial is computed in the time interval of one code word. As a consequence, the original architecture in [2] of a pipeline RS decoder is not as efficient as it might be. A substantial portion of the systolic array is always idling. This fact makes possible a more efficient design with fewer cells and no loss in the throughput rate.

For the (N, I) RS code the length of the syndrome polynomial is $N - I$. The maximum length of the resultant Forney syndrome polynomial is also $N - I$. Imagine now that a single cell is used recursively to perform the successive steps of the modified Euclidean algorithm instead of pipelining data to

the next cell. Then it would take $N - I$ recursions to complete the algorithm, where each recursion requires $N - I$ symbol times. Therefore, using a single cell recursively requires only a total of $(N - I)^2$ symbol time to complete the modified form of Euclidean algorithm. Since a syndrome polynomial needs to arrive every N symbol times, only $\lfloor (N - I)^2/N \rfloor$ cells are needed to process successive syndrome polynomials at a full pipeline throughput rate.

Figure 4 shows the new alternate architectural design. The input multiplexer directs the syndrome polynomials to different cells. Each processor cell is almost identical to the cell presented in [2], except that it is used to process data recursively.

The architecture of the new basic cell is given in Fig. 5. Compared with the previous systolic array design [2], the present scheme for multiplexing the recursive cell computations significantly reduces the number of cells and as a consequence the number of circuits. Table 1 shows that the cell reduction is greater for high rate codes.

VI. A VLSI Design of a Polynomial Evaluation Circuit

In RS decoding the errata locator polynomial

$$\sigma(X) = \sum_{i=0}^{e+E} \sigma_i X^i \tag{23}$$

its derivative

$$\sigma'(X) = \sum_{\substack{i=0 \\ 2 \nmid i}}^{e+E} \sigma_i X^{i-1} \tag{24}$$

and the errata evaluator polynomial

$$\omega(X) = \sum_{i=0}^{e+E-1} \omega_i X^i \tag{25}$$

all need to be evaluated for each α^{-i}, $1 \leqslant i \leqslant N$. Note that the syndrome computation is another form of evaluating the received message polynomial $R(X)$:

$$S_k = R(X)\Big|_{X=\alpha^k} = \sum_{i=0}^{N-1} r_i X^i \Big|_{X=\alpha^k} \quad \text{for } 1 \leqslant k \leqslant d-1 \tag{26}$$

However, the syndrome computation is an evaluation of a polynomial of length N on $d - 1$ points and both $\sigma(X)$ and $\omega(X)$, having length $\leqslant e + E + 1 \leqslant d - 1$, are evaluated on N points. If one evaluates $\sigma(X)$ or $\omega(X)$ using the design in Section III for syndrome computation, it would take N of these cells. Since $N > d - 1$, there is a more efficient design which uses only $d - 1$ cells with less complexity.

Consider evaluating a polynomial $A(X)$, $\deg(A(X)) \leqslant d - 2$

$$A(X) = \sum_{i=0}^{d-2} a_i X^i \tag{27}$$

for $X = \alpha^{-j}$, $j = 1, 2, \ldots, N$.

Hence,

$$A(X)\Big|_{X=\alpha^{-j}} = \sum_{i=0}^{d-2} a_i \alpha^{-ji} = \sum_{i=0}^{d-2} a_i \alpha^{-ij} \quad \text{for } j = 1, 2, \ldots, N \tag{28}$$

For each a_i, the quantity $a_i(\alpha^{-i})^j$ can be obtained by recursively multiplying a fixed constant α^i as j goes from 1 to N.

A finite field summation of $d - 1$ terms results in the desired polynomial evaluation. A systolic array design of such an operation is shown in Fig. 6. Note that the results of evaluating $\sigma(X)$, $\sigma'(X)$, and $\omega(X)$ are produced sequentially. This matches perfectly with the sequential nature of the received data $R(X)$ in a real-time decoding environment.

One last observation on the polynomial evaluation: the evaluation of $\sigma'(X)$ uses only the coefficients of $\sigma(X)$ with odd power terms. This property makes it possible to obtain the evaluation of $\sigma'(X)$ as a by-product from the evaluation of $\sigma(X)$ at no cost. As illustrated in Fig. 7, simply use two smaller exclusive-OR trees to sum the even terms and odd terms of $\sigma(X)$ separately. The summation of the odd terms yields $\sigma'(\alpha^{-i})$. Another exclusive-OR operation on the two partial sums results in $\sigma(\alpha^{-i})$ itself.

References

[1] 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.

[2] R. P. Brent and H. T. Kung, "Systolic VLSI Arrays for Polynomial GCD Computations," Dept. Computer Science, Carnegie-Mellon Univ., Pittsburgh, Pennsylvania, 1982.

[3] 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*, October–December 1985, Jet Propulsion Laboratory, Pasadena, California, pp. 73–81, February 15, 1986.

[4] I. S. Reed, T. K. Truong, and R. L. Miller, "Decoding of BCH and RS Codes with Errors and Erasures Using Continued Fractions," *Electronic Letters*, vol. 15, no. 17, pp. 542–544, July 1979.

[5] T. K. Truong, I. S. Hsu, I. S. Reed, and W. L. Eastman, "Simplified Procedure for Correcting Both Errors and Erasures of a Reed–Solomon Code Using the Euclidean Algorithm," *TDA Progress Report 42-91*, July–September 1987, Jet Propulsion Laboratory, Pasadena, California, November 15, 1987.

[6] T. K. Truong, I. S. Hsu, I. S. Reed, L. J. Deutsch, E. Satorius, and H. Shao, "A Comparison of the VLSI Architecture for Time and Transform Domain Decoding of Reed-Solomon Codes," to appear in *Proc. ICCD '87*, Port Chester, New York, October 5-8, 1987.

[7] R. J. McEliece, *The Theory of Information and Coding*, Reading, Massachusetts: Addison-Wesley Publishing Company, 1977.

Table 1. Comparison of the number of cells required in the modified Euclid's algorithm computation

RS code	Full systolic array	Multiplexing on recursive cells
(15, 9)	6	3
(31, 15)	16	9
(255, 223)	32	5

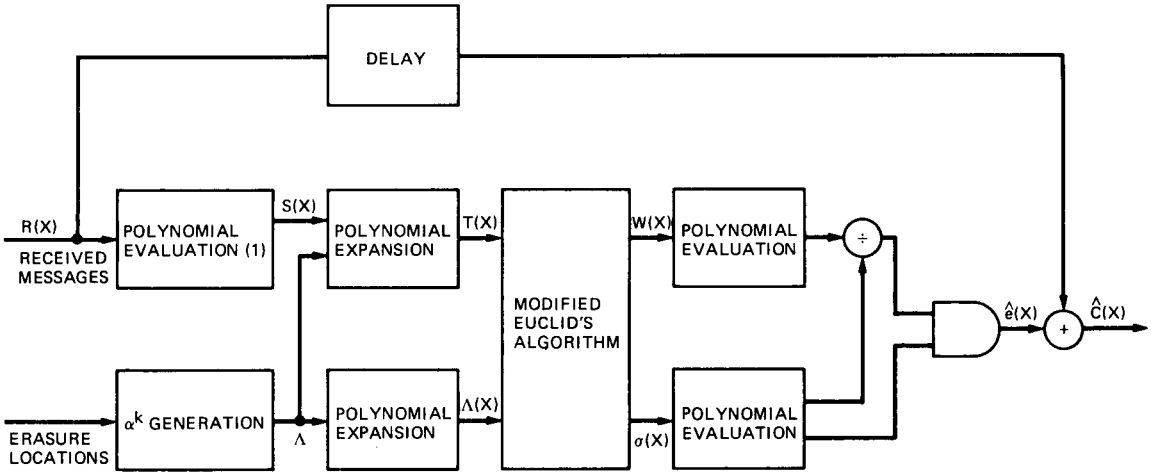

Fig. 1. VLSI architecture of a pipeline time-domain Reed-Solomon decoder for both error and erasure correction

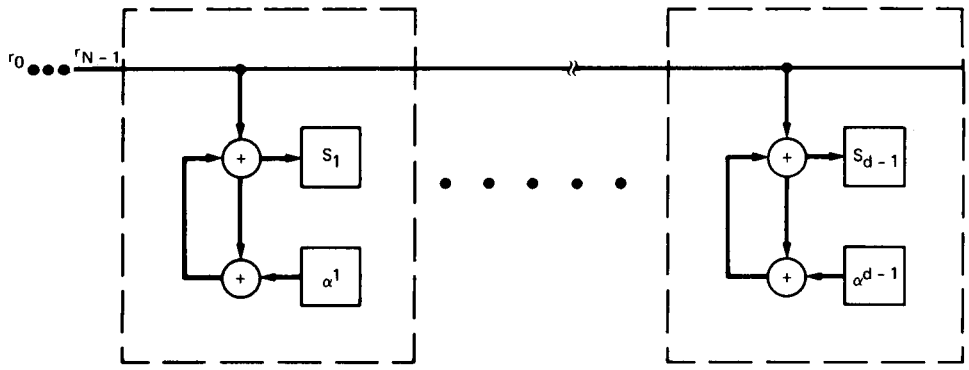

Fig. 2. A systolic array to compute syndromes

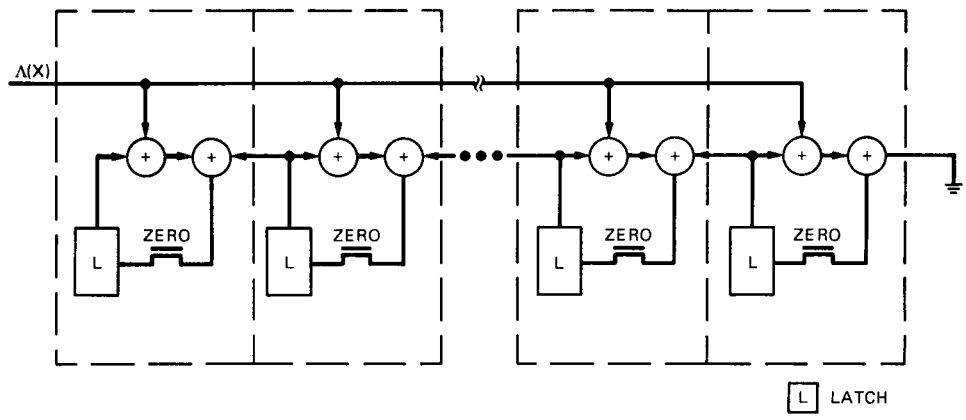

Fig. 3. A systolic array for polynomial expansion computation

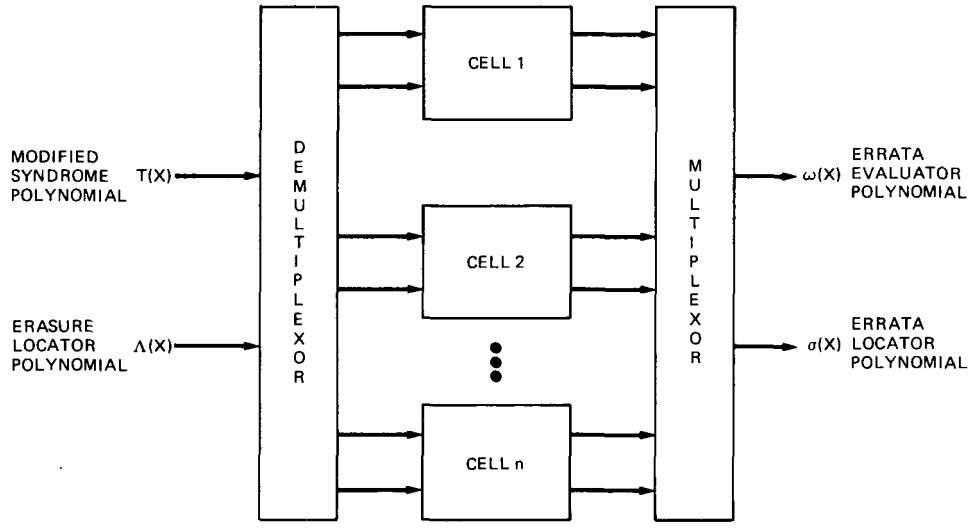

Fig. 4. The new architecture for performing the modified form of Euclid's algorithm

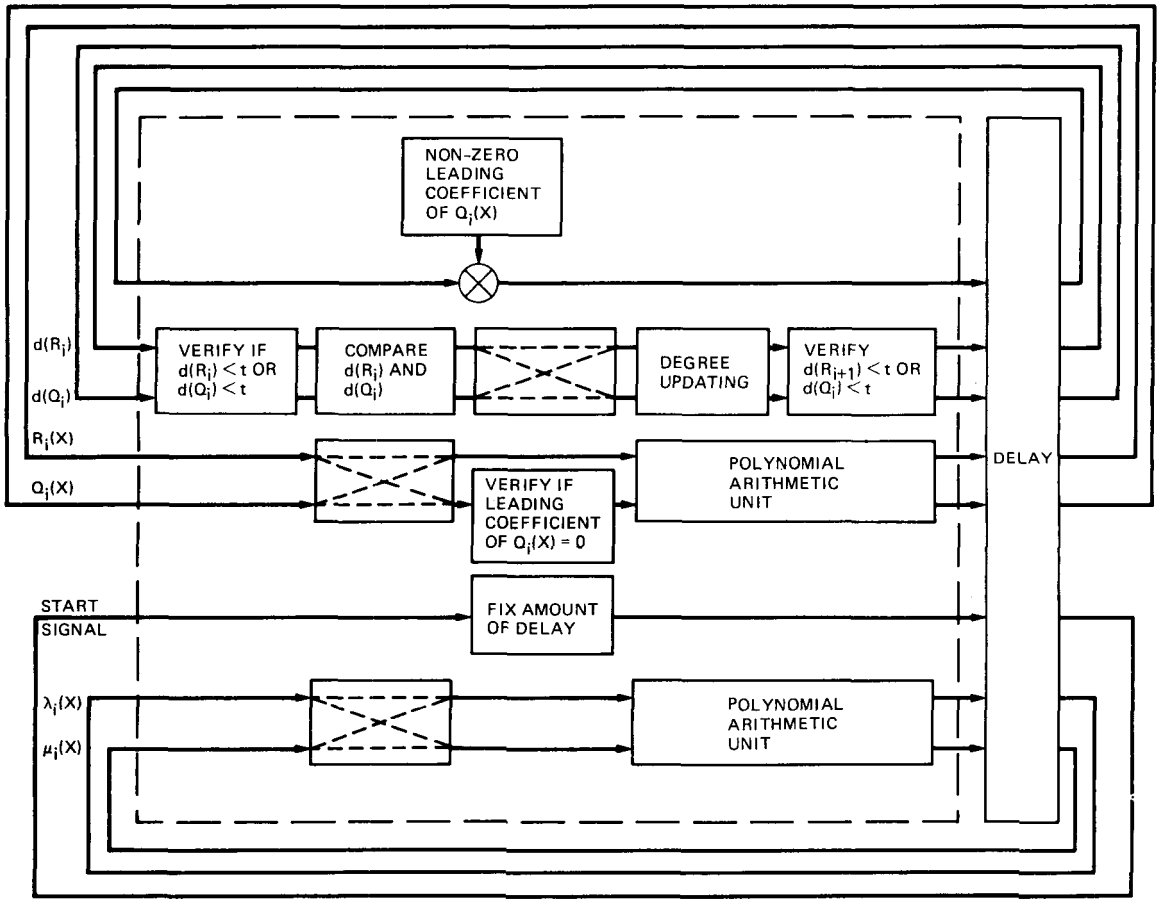

Fig. 5. Block diagram of basic cell for computing the modified Euclid's algorithm

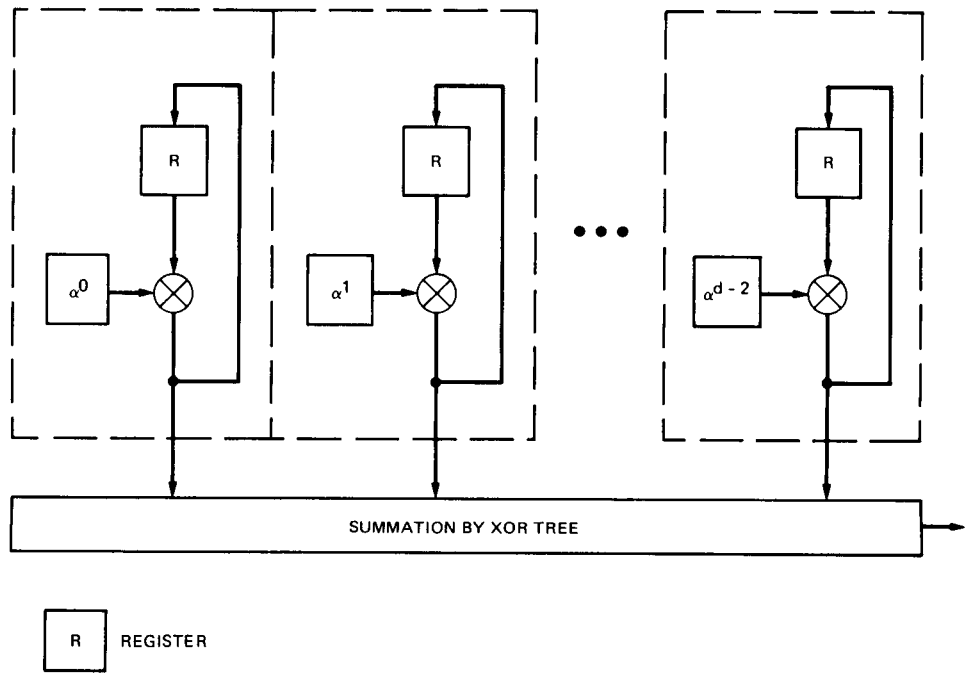

Fig. 6. A systolic array for polynomial evaluation

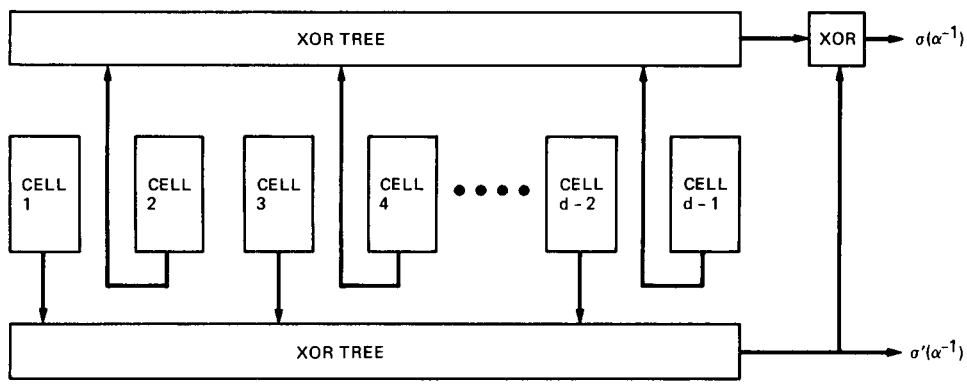

Fig. 7. The polynomial evaluation circuit for $\sigma(X)$ and $\sigma'(X)$