A Systolic VLSI Design of a Pipeline Reed-Solomon Decoder

H. M. Shao, T. K. Truong, L. J. Deutsch, and J. H. Yuen
Communication System Research Section

I. S. Reed
University of Southern California

A pipeline structure of a transform decoder similar to a systolic array is developed to decode Reed-Solomon (RS) codes. An important ingredient of this design is a modified Euclidean algorithm for computing the error locator polynomial. The computation of inverse field elements is completely avoided in this modification of Euclid's algorithm. The new decoder is regular and simple, and naturally suitable for VLSI implementation. An example illustrating both the pipeline and systolic array aspects of this decoder structure is given for a (15, 9) RS code.

I. Introduction

A concatenated coding system consisting of a convolutional inner code and a Reed-Solomon (RS) outer code has been adopted as the standard for future space missions by both the European Space Agency (ESA) and NASA (Ref. 1). The convolutional inner code is a (7, 1/2) code which is also used on the Voyager Project. The outer Reed-Solomon code is a (255, 223) block code of 8-bit symbols that is capable of correcting up to 16 symbol errors. The performance of such a concatenated code is investigated in Ref. 2 where it is shown that the concatenated channel provides a coding gain of approximately 2 dB over the channel with only convolutional coding for a decoded bit error rate of 10^{-5}.

Some work has already been done in developing VLSI encoders for RS codes. The algorithm that was developed in Ref. 3 by Berlekamp was used in Ref. 4 to design a single chip (255, 223) RS encoder.

Recently it was suggested by Brent and Kung (Ref. 5) that a pipeline architecture could be used to compute the greatest common divisor (GCD) of two polynomials. In Euclid's algorithm, the GCD of two polynomials can be used to obtain the error locator polynomial of an RS code. It is shown in this article that a modified form of Euclid's algorithm, based on the idea of Brent and Kung, can be used to find the error locator polynomial in a transform RS decoder. This algorithm requires no multiplications by an inverse element in the finite field.

Utilizing the above mentioned modification of Euclid's algorithm, a new pipeline architecture for a transform decoder is developed to decode RS codes. This architecture can be implemented on VLSI chips with nMOS technology. Such a pipeline decoding algorithm can be realized in a systolic-array architecture, which is presented in this article as an example.

II. A VLSI Design of a RS Decoder

Let $GF(2^m)$ be the 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 $d = 2t + 1$, where $I = N - (d - 1)$ denotes the number of m-bit message symbols and t denotes the number of errors that may be corrected in each codeword.

The standard transform decoder for decoding RS codes was described in Ref. 6. In the transform decoding algorithm, the Berlekamp-Massey, Euclid, or continued-fraction algorithm can be used to find the error locator polynomial. It was decided for this design to use the Euclid algorithm because its modularity makes it well suited for VLSI implementation. The operations needed to compute Euclid's algorithm generally require the computation of inverse elements in GF(2^m). To avoid the computation of inverse elements, a modification of Euclid's algorithm can be utilized to find the error locator polynomial.

The transform decoding algorithm, which utilizes the modified form of Euclid's algorithm to find the error locator polynomial, is described in the following four steps:

Step 1: Compute the syndromes

$$S_k = \sum_{n=0}^{N-1} r_n \alpha^{nk} \quad , \quad 1 \leqslant k \leqslant 2t \tag{1}$$

where r_n $(0 \leqslant n \leqslant N - 1)$ is the received code pattern. Note that $E_k = S_k$ $(1 \leqslant k \leqslant 2t)$, where E_k is the transform of the error sequence (Ref. 5).

Step 2: Perform the modified Euclid algorithm (described in the next section) on x^{2t} and the syndrome polynomial

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

to obtain the polynomial,

$$\lambda(x) = \lambda_0 x^t + \lambda_1 x^{t-1} + \cdots + \lambda_t \tag{3a}$$

Divide the polynomial $\lambda(x)$ by its leading coefficient λ_0. This yields the standard error-locator polynomial.

$$\sigma(x) = x^t + \sigma_1 x^{t-1} + \cdots + \sigma_t \tag{3b}$$

where

$$\sigma_i = \frac{\lambda_i}{\lambda_0}, \quad 1 \leqslant i \leqslant t.$$

Step 3: From the known coefficients σ_k in Eq. (3b), compute the remaining elements E_k of the transform of the error sequence for $d \leqslant k \leqslant N$ where $E_0 = E_N$ from the equation (Ref. 5):

$$E_{2t+j} + \sum_{k=1}^{2t} (-1)^k \sigma_k E_{2t+j-k} = 0 \quad \text{for } j \geqslant 1 \tag{4}$$

Step 4: Compute the inverse transform of E_k over $GF(2^m)$ to obtain the estimated error pattern. That is,

$$e_n = \sum_{k=0}^{N-1} E_k \alpha^{-nk}, \quad 0 \leqslant n \leqslant N-1 \tag{5}$$

Finally, the estimate of the original code vector is obtained by subtracting the error pattern from the received pattern.

Figure 1 shows an overall block diagram of this decoder. In the following sections, a VLSI design for each of the functional blocks is described.

III. A VLSI Design for Computing Syndromes

In this section, a VLSI architecture is developed to compute the syndromes for an (N, I) RS code over $GF(2^m)$. The decoding procedure and the VLSI architecture is illustrated in the following simple example:

Example. Consider a (15, 9) RS code over $GF(2^4)$ with minimum distance $d = 7$. In this code, $t = 3$ errors can be corrected.

To compute a generator polynomial for this code (Ref. 7), one can use the representation of the field $GF(2^4)$ given in Table A-1 of the Appendix. One generator polynomial of such a (15, 9) RS code is

$$g(x) = \prod_{i=1}^{6} (x - \alpha^i)$$

$$= x^6 + \alpha^{10} x^5 + \alpha^{14} x^4 + \alpha^4 x^3 + \alpha^6 x^2 + \alpha^9 x + \alpha^6 \tag{6}$$

Assume the message symbols are

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

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

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

Written as a vector, the codeword is

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

Suppose that two errors exist in the error pattern as follows:

$$e(x) = (0,0,0,0,\alpha^{11},0,0,0,0,0,0,\alpha^{7},0,0,0) \quad (9)$$

Then the received pattern is

$$r(x) = (r_{14}, r_{13}, \cdots, r_0) = c(x) + e(x)$$
$$= (\alpha^{10}, \alpha^{12}, \alpha^{8}, \alpha^{5}, \alpha, \alpha^{14}, \alpha^{13}, \alpha^{11}, \alpha^{9}, \alpha^{0}, \alpha, \alpha^{12}, \alpha^{6}, \alpha^{12}, \alpha^{8}) \quad (10)$$

The syndromes of this received message are

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

This yields $S_1 = \alpha^7$, $S_2 = \alpha^{12}$, $S_3 = \alpha^6$, $S_4 = \alpha^{12}$, $S_5 = \alpha^{14}$ and $S_6 = \alpha^{14}$. To develop a VLSI design to compute S_k, let the first expression in Eq. (11) be rewritten in recursive form as

$$S_k = (\ldots((r_{14}\alpha^{k} + r_{13})\alpha^{k} + r_{12})\alpha^{k} + \cdots)\alpha^{k} + r_0$$
$$= (\ldots((\alpha^{10}\alpha^{k} + \alpha^{12})\alpha^{k} + \alpha^{8})\alpha^{k} + \cdots)\alpha^{k} + \alpha^{8} \quad (12)$$

A structure for computing Eq. (12) is shown in Fig. 2. In this figure, the function of each cell is given by the register transfer relation

$$B_k \leftarrow A_k + B_k\alpha^{7-k} \qquad \text{for } 1 \leqslant k \leqslant 6 \quad (13)$$

where "$\leftarrow$" denotes the operation, "is replaced by." The input data are sent to all of the cells simultaneously. Assume initially that all registers are set to zero. After the complete received codeword is entered, the desired syndromes S_k are contained in registers B_{7-k} $(1 \leqslant k \leqslant 6)$. The syndromes computed in this manner are shifted serially from the right end of register B_6 and fed into the next stage, which performs a modification of Euclid's algorithm. The sequence $\{S_k\}$ of syndromes are conveniently represented as in Eq. (2) by what is called the syndrome polynomial,

$$S(x) = \sum_{k=1}^{2t} S_k x^{2t-k}$$

In particular, the syndrome polynomial for this example is

$$S(x) = \alpha^{7}x^{5} + \alpha^{12}x^{4} + \alpha^{6}x^{3} + \alpha^{12}x^{2} + \alpha^{14}x + \alpha^{14} \quad (14)$$

IV. A Systolic Array Architecture to Compute the Error-Locator Polynomial with a Modified Form of Euclid's Algorithm

It is shown (Ref. 7) that the error locator polynomial can be calculated from the syndrome polynomial by means of Euclid's algorithm. Since quotients are needed in the usual form of Euclid's algorithm, inverses of field elements are required at each stage. The successive computation of these inverses is difficult to realize in a VLSI circuit. It was shown in Ref. 5 that Euclid's algorithm could be modified to eliminate the computation of inverse elements. In this section, this new idea is applied to the computation of the polynomial $\lambda(x)$, defined in Eq. (3a).

Consider the two polynomials,

$$A(x) = x^{2t}$$

$$S(x) = \sum_{k=1}^{2t} S_k x^{2t-k}$$

The modified form of Euclid's algorithm is a recursive procedure for finding the i^{th} remainder $R_i(x)$ and the quantities $\gamma_i(x)$ and $\lambda_i(x)$ that satisfy

$$\gamma_i(x)A(x) + \lambda_i(x)S(x) = R_i(x) \tag{15}$$

When the degree of the remainder $R_i(x)$ is less than t, the algorithm stops. The resulting $\lambda_i(x)$ at the termination of the algorithm is the desired polynomial, $\lambda(x)$, in Eq. (3a).

The initial conditions of the algorithm are

$$R_0(x) = A(x) \qquad Q_0(x) = S(x) \tag{16a}$$

$$\lambda_0(x) = 0 \qquad \mu_0(x) = 1 \tag{16b}$$

$$\gamma_0(x) = 1 \qquad \eta_0(x) = 0 \tag{16c}$$

For $i \geqslant 1$, compute recursively

$$R_i(x) = b_{i-1}[\sigma_{i-1}R_{i-1}(x) + \bar{\sigma}_{i-1}Q_{i-1}(x)] - a_{i-1}x^{1\,i-1}[\sigma_{i-1}Q_{i-1}(x) + \bar{\sigma}_{i-1}R_{i-1}(x)] \tag{17a}$$

$$\lambda_i(x) = b_{i-1}[\sigma_{i-1}\lambda_{i-1}(x) + \bar{\sigma}_{i-1}\mu_{i-1}(x)] - a_{i-1}x^{1\,i-1}[\sigma_{i-1}\mu_{i-1}(x) + \bar{\sigma}_{i-1}\lambda_{i-1}(x)] \tag{17b}$$

$$\gamma_i(x) = b_{i-1}[\sigma_{i-1}\gamma_{i-1}(x) + \bar{\sigma}_{i-1}\eta_{i-1}(x)] - a_{i-1}x^{1\,i-1}[\sigma_{i-1}\eta_{i-1}(x) + \bar{\sigma}_{i-1}\gamma_{i-1}(x)] \tag{17c}$$

$$Q_i(x) = \sigma_{i-1}Q_{i-1}(x) + \bar{\sigma}_{i-1}R_{i-1}(x) \tag{17d}$$

$$\mu_i(x) = \sigma_{i-1}\mu_{i-1}(x) + \bar{\sigma}_{i-1}\lambda_{i-1}(x) \tag{17e}$$

$$\eta_i(x) = \sigma_{i-1}\eta_{i-1}(x) + \bar{\sigma}_{i-1}\gamma_{i-1}(x) \tag{17f}$$

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

$$1_{i-1} = \deg(R_{i-1}(x)) - \deg(Q_{i-1}(x))$$

$$\sigma_{i-1} = 1 \qquad \text{if } 1_{i-1} \geqslant 0$$

$$\sigma_{i-1} = 0 \qquad \text{if } 1_{i-1} < 0$$

The algorithm stops when $\deg(R_i(x)) < t$.

The proof that this algorithm performs the desired computation is similar to that of the standard version of Euclid's algorithm, and it is not included here. Notice that the $2t$ computations of inverse field elements that are needed in the usual form of Euclid's algorithm are not required in the above modified version of Euclid's algorithm. The computational details needed to find the error-locator polynomial of an RS(15, 9) code over $GF(2^4)$ using the modified Euclid's algorithm are given next in a continuation of the previous example. The implementation of this algorithm is accomplished by means of a systolic array cell design. Such a systolic array structure is presented here in considerable detail.

Consider the decoding of the three-error-correcting RS(15, 9) code described previously. The syndrome polynomial $S(x)$, given in Eq. (14), is used in the modified Euclid's algorithm to calculate $\lambda(x)$.

Step 1. By Eq. (17), for $i = 1$: $1_0 = 6 - 5 = 1$ and $\sigma_0 = 1$. Hence let

$$R_1(x) = \alpha^7 R_0(x) + xQ_0(x)$$

$$\lambda_1(x) = \alpha^7\lambda_0(x) + x\mu_0(x)$$

$$Q_1(x) = Q_0(x) \text{ and } \mu_1(x) = \mu_0(x)$$

where

$$R_0(x) = x^6, Q_0(x) = S(x)$$

$$\lambda_0(x) = 0 \text{ and } \mu_0(x) = 1$$

Note that $\gamma_i(x)$ and $\eta_i(x)$ are not needed in the computation of $\lambda(x)$. Thus,

$$R_1(x) = \alpha^{12}x^5 + \alpha^6x^4 + \alpha^{12}x^3 + \alpha^{14}x^2 + \alpha^{14}x$$

$$\lambda_1(x) = x, Q_1(x) = S(x), \text{and } \mu_1(x) = 1$$

To implement this operation, let the coefficients of $R_0(x)$, $Q_0(x)$, $\lambda_0(x)$ and $\mu_0(x)$ enter the first cell of the systolic array in the manner shown in Fig. 3.

The quantities $d(R_0)$ and $d(Q_0)$ are integers representing the degrees of $R_0(x)$ and $Q_0(x)$, respectively. The "start" signal is a timing signal used to indicate the beginning of the polynomials, i.e., the leading coefficients. Note that to compute $R_1(x) = \alpha^7 R_0(x) + xQ_0(x)$ from $R_0(x)$ and $Q_0(x)$ term by term, the leading coefficients of $R_0(x)$ and $Q_0(x)$ are aligned with the start signal. Such an arrangement corresponds to the simultaneous entry of $R_0(x)$ and $xQ_0(x)$ into cell 1. The inputs $\lambda_0(x)$ and $x\mu_0(x)$ to cell 1 are aligned with the start signal in a similar manner.

The functional block diagram of cell 1 is shown in Fig. 4(a). The start signal, as well as $xQ_0(x)$ and $x\mu_0(x)$, are delayed by one time unit in such a manner that the leading coefficients of $R_1(x)$, $Q_1(x)$, $\lambda_1(x)$ and $\mu_1(x)$ are properly initiated by the start signal at the output of cell 1.

Step 2. Since $1_1 = \deg(R_1(x)) - \deg(Q_1(x)) = 0, \sigma_1 = 1$. Thus,

$$R_2(x) = \alpha^7 R_1(x) + \alpha^{12} Q_1(x)$$

$$= \alpha^{10}x^4 + \alpha^7 x^3 + \alpha^5 x^2 + \alpha + \alpha^{11}$$

$$\lambda_2(x) = \alpha^7 \lambda_1(x) + \alpha^{12}\mu_1(x) = \alpha^7 x + \alpha^{12}$$

$$Q_2(x) = Q_1(x) \text{ and } \mu_2(x) = \mu_1(x)$$

The implementation of cell 2, shown in Fig. 4(b), is essentially identical to cell 1.

Step 3. From cell 2, $d(R_2) = 4$ and $d(Q_2) = 5$. Thus,

$$1_2 = -1 < 0 \text{ and } \sigma_2 = 0$$

Hence, by Eq. (17),

$$R_3(x) = \alpha^{10} Q_2(x) + \alpha^7 x R_2(x)$$

$$= \alpha x^4 + \alpha^{13} x^3 + \alpha^{11} x^2 + \alpha x + \alpha^9$$

$$\lambda_3(x) = \alpha^{10}\mu_2(x) + \alpha^7 x \lambda_2(x)$$

$$= \alpha^{14}x^2 + \alpha^4 x + \alpha^{10}$$

$$Q_3(x) = R_2(x) \text{ and } \mu_3(x) = \lambda_2(x)$$

In Fig. 4(c), the role-switching operation is implemented by a simple crossover at the inputs of cell 3. Following the switching operation, cell 3 is identical in operation to cells 1 and 2.

Step 4. Since $d(R_3) = 4$ and $d(Q_3) = 4$, $1_3 = 0$ and $\sigma_3 = 1$. Thus,

$$R_4(x) = \alpha^{12} R_3(x) + \alpha Q_3(x) = 0x^3 + 0x^2 + \alpha^9 x + \alpha^6$$

$$\lambda_4(x) = \alpha^{10}\lambda_3(x) + \alpha\mu_3(x) = \alpha^9 x^2 + \alpha^6 x + \alpha^7$$

$$Q_4 = Q_3(x) \text{ and } \mu_4(x) = \mu_3(x)$$

The implementation of this step is presented in Fig. 4(d), which is again identical to cells 1 and 2. Note that $d(R_4) = 3$ although the actual degree of $R_4(x)$ is only 1. The reason for this is that $R_4(x)$ can be viewed as a polynomial of degree 3 even though the coefficients of x^3 and x^2 are both zero. Such an arrangement reduces the degree of polynomials regularly by one as the polynomials propagate from cell to cell. At the next stage one will see how zero leading coefficients are treated.

Step 5. Since $d(R_4) = 3$ and $d(Q_4) = 4$, $1_4 = 3 - 4 = -1$ and $\sigma_4 = 0$. However, the computation cannot continue to obtain $R_5(x)$ from $R_4(x)$ and $Q_4(x)$, term-by-term, since the leading coefficient of $R_4(x)$ is zero. Hence the proper treatement is to first assign

$$R_5(x) = Q_4(x), Q_5(x) = R_4(x)$$

$$\lambda_5(x) = \mu_4(x), \mu_5(x) = \lambda_4(x)$$

and to then reduce the degree of $Q_5(x)$ by one and pass the result on to the next cell. That is, let

$$Q_5(x) = 0x^2 + \alpha^9 x + \alpha^6 \text{ and } d(Q_5) = 3 - 1 = 2$$

In this particular example note that after the reduction, $d(Q_5) = 2 < t = 3$. Hence the systolic array should stop calculation at this point and accept $\mu_5(x)$ as the final result, namely,

$$\lambda(x) = \mu_5(x) = \alpha^9 x^2 + \alpha^6 x + \alpha^7 \qquad (18)$$

The functional diagram for the implementation of this step is shown in Fig. 4(e). At the output, the start signal's 1 points to the leading coefficient of 0 of $\lambda(x) = 0x^3 + \alpha^9 x^2 + \alpha^6 x + \alpha^7$.

Since the syndrome polynomial can have degree at most $2t - 1$ and each cell in this modified algorithm reduces the degree of one of the input polynomials by one, it will take $2t$ cells in the worst case to produce an output polynomial of degree less than t. Hence, only $2t$ such cells are needed in the RS decoder. If the algorithm actually concludes after the i^{th} cell, then the $2t - i$ subsequent cells will pass the polynomials unchanged as did the 6^{th} cell in the above example (see Fig. 4(f).

The function for a typical cell of this array is described by the flowchart in Fig. 5. The architecture of the cell is given in Fig. 6. The above systolic array design has been simulated completely on a computer, and it has been validated.

V. The Recursive Circuit for Computing the Remaining Transform Error Pattern

To compute the remaining transform error pattern of an (15, 9) RS code, Eq. (18) is multiplied by α^{-9} to yield

$$\alpha(x) = x^2 + \sigma_1 x + \sigma_2$$

where

$$\sigma_1 = \alpha^{12} \text{ and } \sigma_2 = \alpha^{13}$$

Thus, by Eq. (4), the recursive equation is

$$E_{j+2} = \alpha^{12} E_{j+1} + \alpha^{13} E_j \; (j = 5, \cdots, 13) \qquad (19)$$

The circuit for implementing Eq. (19) is shown in Fig. 7. In Fig. 7, registers R_1 and R_2 are initialized to contain E_5 and E_6, respectively. Thus $E_7 = \alpha^0$, $E_8 = 0$, $E_9 = \alpha^{13}$, $E_{10} = \alpha^{10}$, $E_{11} = \alpha^8$, $E_{12} = \alpha^4$, $E_{13} = \alpha^{11}$, $E_{14} = \alpha^0$, and $E_{15} = E_0 = \alpha^8$ are obtained sequentially. The same circuit extends naturally to the general case given in Eq. (4) for an (N, I) RS decoder over $GF(2^m)$.

VI. A VLSI Design for Computing the Error Pattern

By Eq. (5), the inverse transform of the error pattern obtained in the last section is

$$e_k = \sum_{n=0}^{14} E_n \alpha^{-nk}, \quad 0 \leqslant k \leqslant 14$$

or

$$e_k = (\ldots((((E_{14}\alpha^k + E_{13})\alpha^k + E_{12})\alpha^k + \cdots)\alpha^k + \cdots E_0)$$

$$= (\ldots(((\alpha^0 \alpha^k + \alpha^{11})\alpha^k + \alpha^4)\alpha^k \cdots)\alpha^k \cdots)\alpha^k + \cdots \alpha^8) \qquad (20)$$

The VLSI technique for computing Eq. (20) in a (15, 9) RS code over $GF(2^4)$ is similar to that used in the VLSI design for computing syndromes. The number of basic cells needed to compute Eq. (20) is 15. The pipeline structure for computing Eq. (20) is given in Figure 8. The desired result is given in Eq. (9). The same circuit extends to a VLSI design for computing Eq. (5) in a general (N, I) RS code over $GF(2^m)$.

References

1. Kummer, H., *Recommendation for Space Data System Standards: Telemetry Channel Coding*, Issue-1, Consultative Committee for Space Data Systems, Panel 1: Telemetry, Tracking, Command, September, 1983.

2. Miller, R. L., Deutsch, L. J., and Butman, S. A., *On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes*, Publication 81-9, Jet Propulsion Laboratory, Pasadena, California, September 1, 1981.

3. Berlekamp, E. R., "Bit-Serial Reed-Solomon Encoders," *IEEE Transactions on Information Theory*, Vol. IT-28, Number 6, November 1982.

4. Truong, T. K., Deutsch, L. J., Reed, I. S., Hsu, J. S., Wang, K., Yeh, C. S., "The VLSI Design of a Reed-Solomon Encoder Using Berlekamp's Bit Serial Algorithm," *Proceedings of the Third Caltech Conference on VLSI*, California Institute of Technology, Pasadena, California, 1983.

5. Brent, R. P., and Kung, H. T., *Systolic VLSI Arrays for Polynomial GCD Computations*, CMU Computer Science Department Report, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1982.

6. Reed, I. S., Scholtz, R. A., Truong, T. K., and Welch, L. R., "The Fast Decoding of Reed-Solomon Codes Using Fermat Theoretic Transforms and Continued Fractions," *IEEE Transactions on Information Theory*, Vol. IT-24, pp. 100-106, 1978.

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

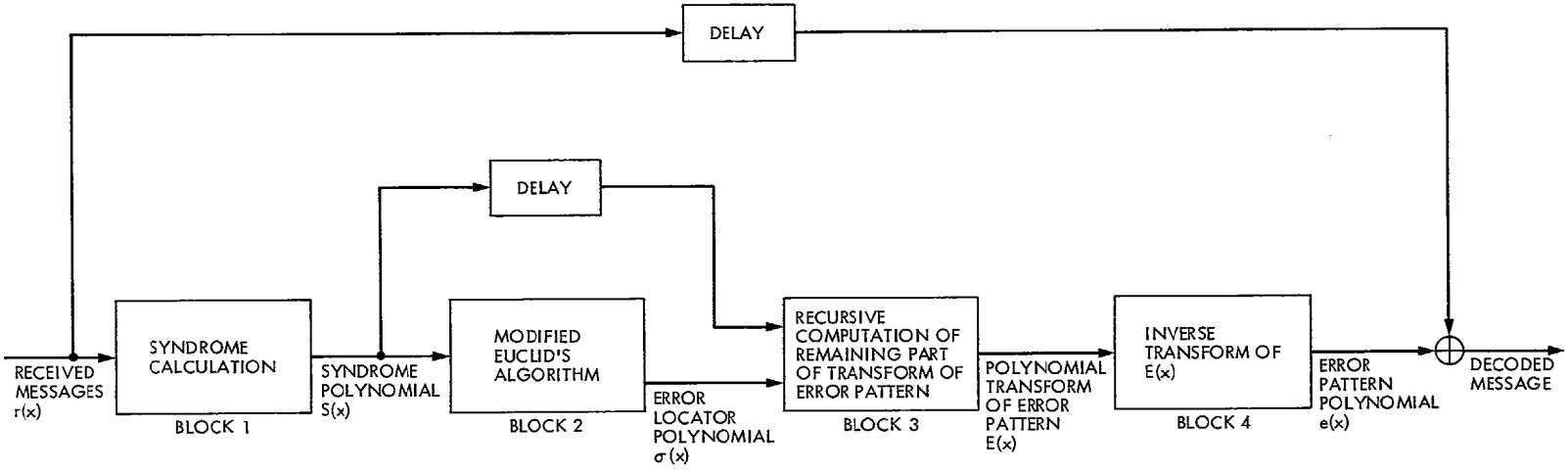

Fig. 1. The overall block diagram of A pipeline (*N*, *I*) RS decoder

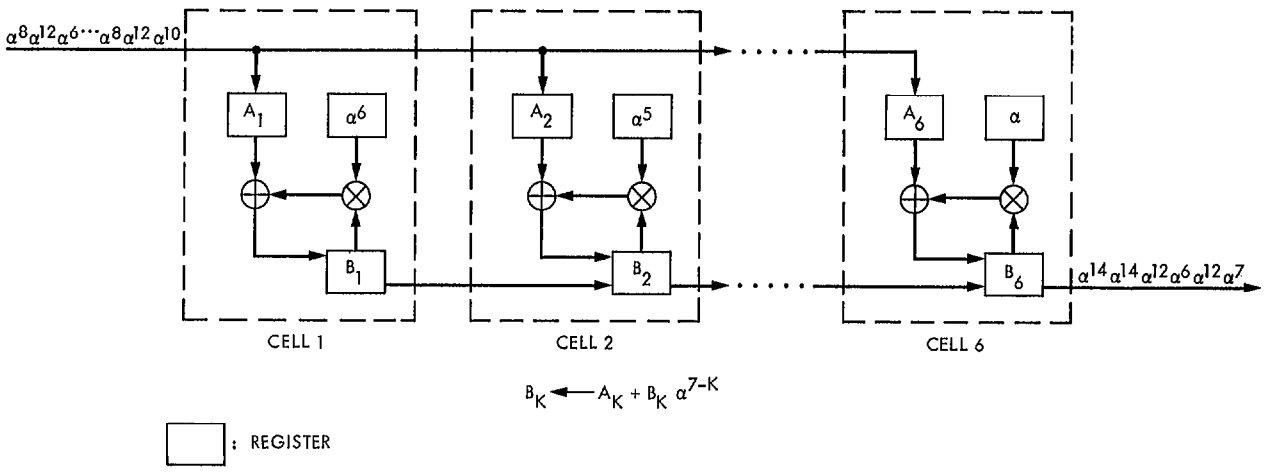

Fig. 2. The systolic array to compute syndromes of a (15, 9) RS code over *GF*(2^4)

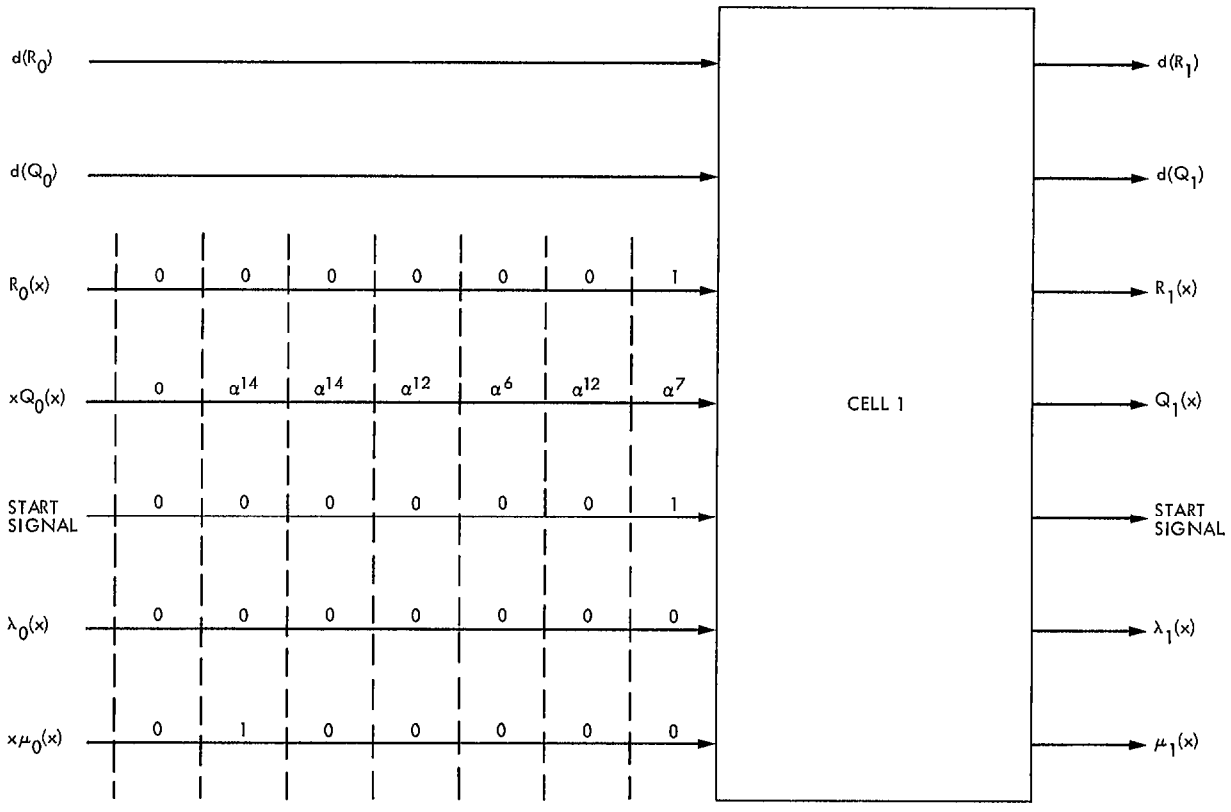

Fig. 3. The timing sequence of the input data

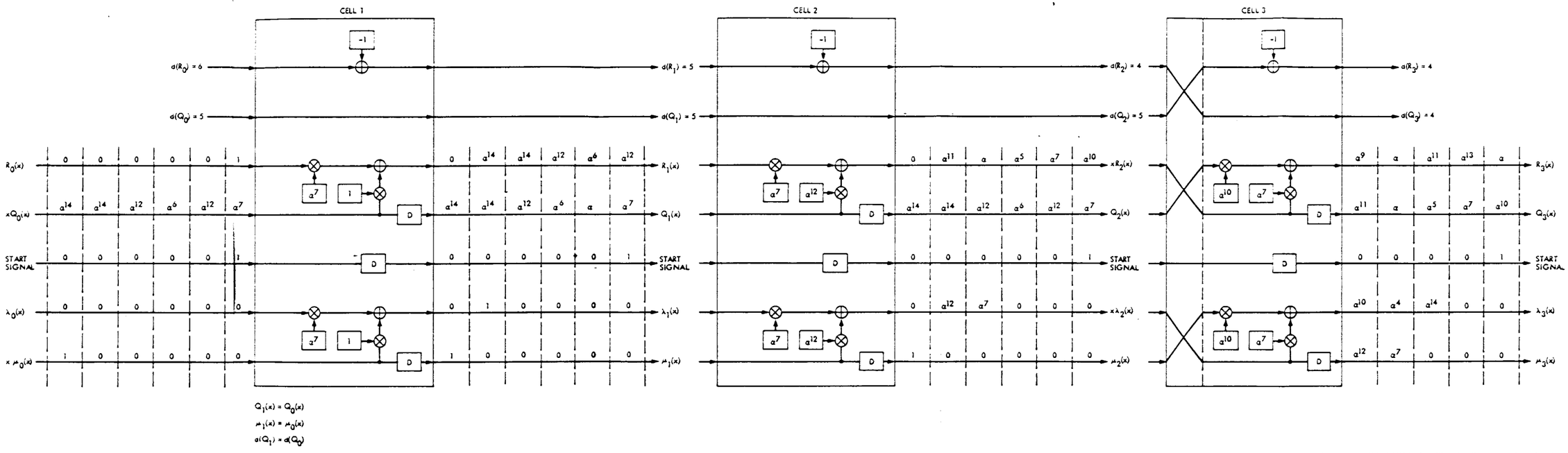

Fig. 4. The modified GCD algorithm for the example in the text

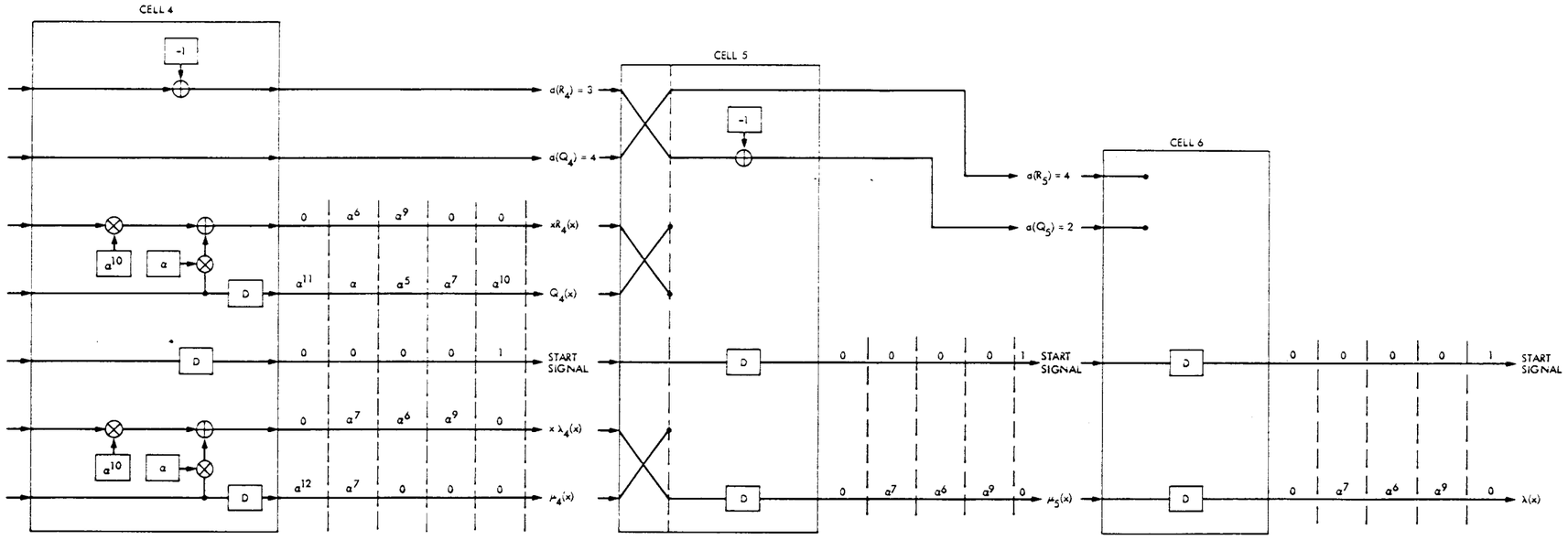
CELL 4
CELL 5
CELL 6
-1
D
a^{10}
α
$a(R_4) = 3$
$a(Q_4) = 4$
$xR_4(x)$
$Q_4(x)$
$x\lambda_4(x)$
$\mu_4(x)$
START SIGNAL
$a(R_5) = 4$
$a(Q_5) = 2$
$\mu_5(x)$
$\lambda(x)$
0 a^6 a^9 0 0
a^{11} α a^5 a^7 a^{10}
0 0 0 0 1
0 a^7 a^6 a^9 0
a^{12} a^7 0 0 0

Fig. 4 (contd)

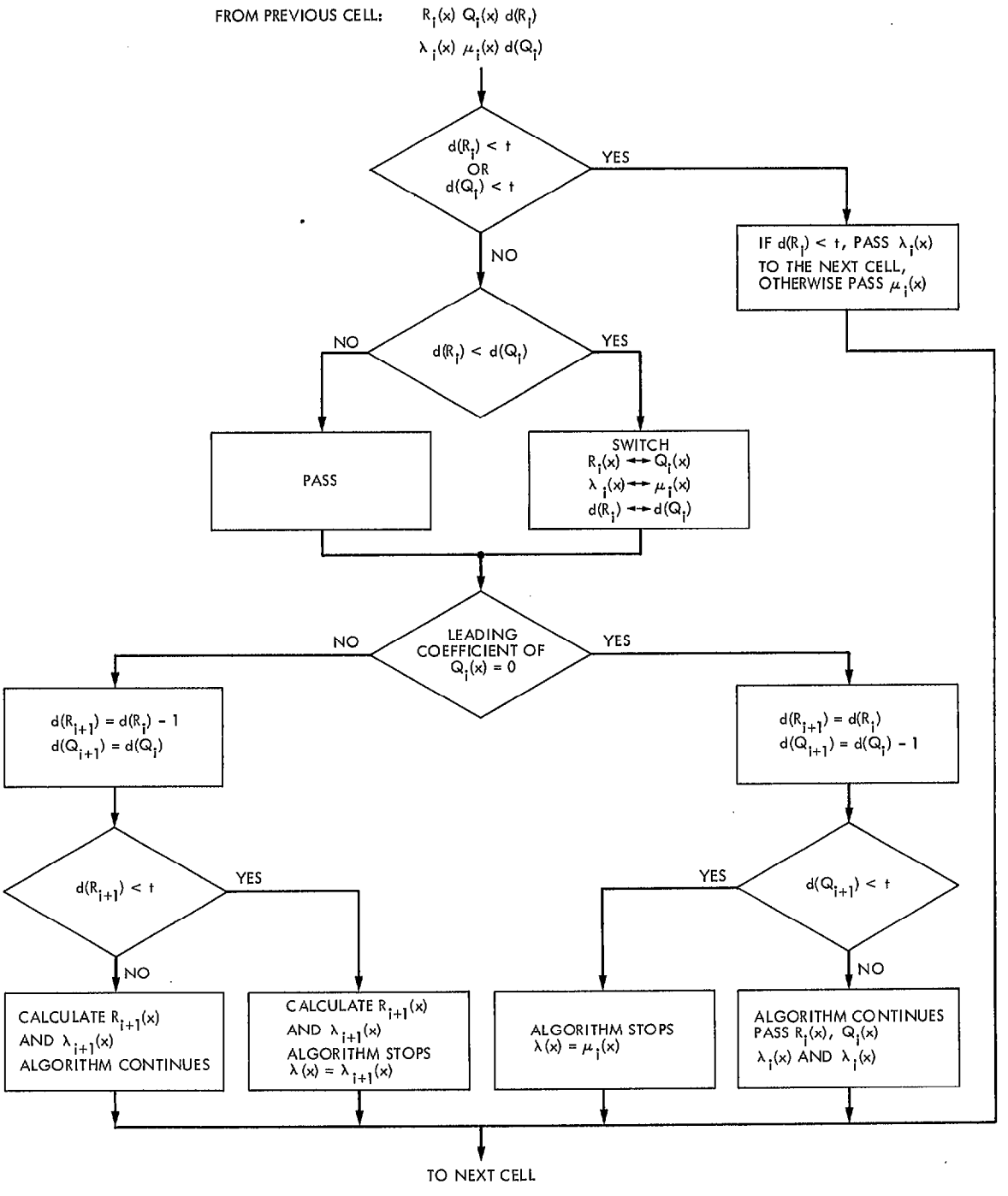

Fig. 5. The flowchart of the function of each cell

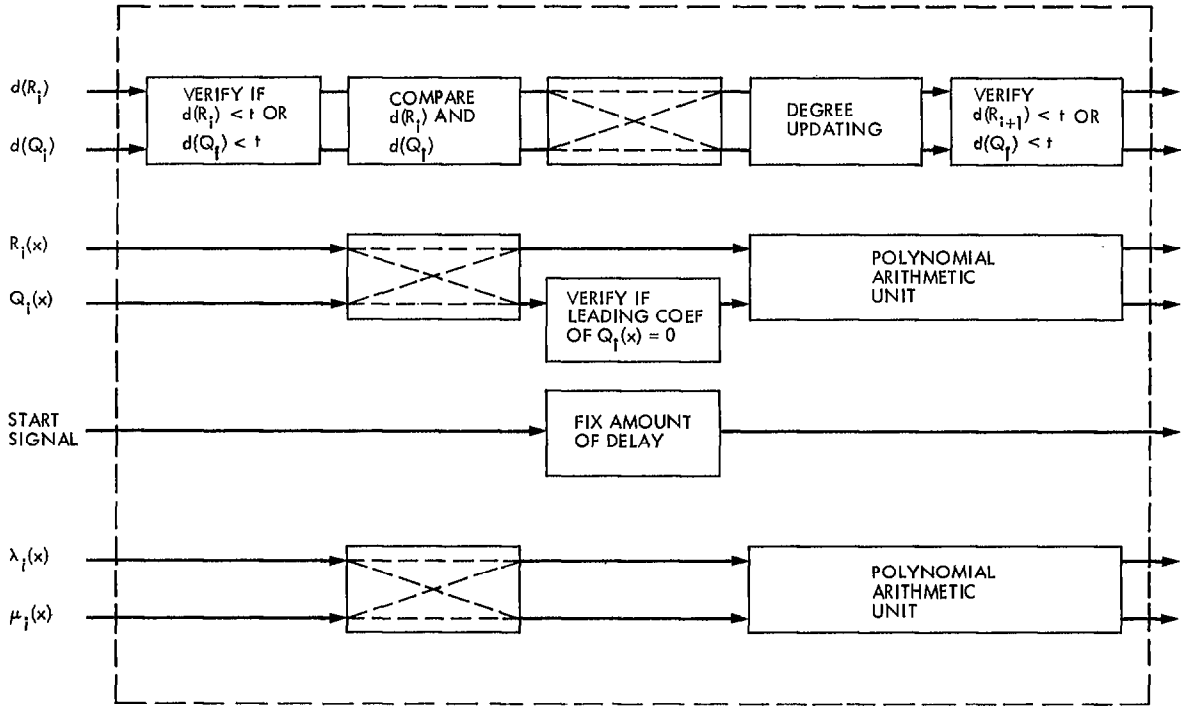

Fig. 6. The architecture of one cell of a systolic array to compute polynomial with modified Euclid's algorithm

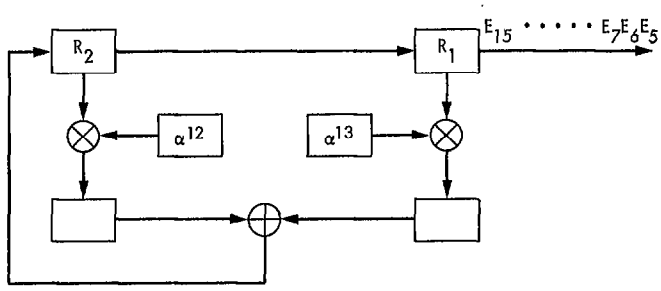

Fig. 7. A recursive circuit for computing the transform error pattern

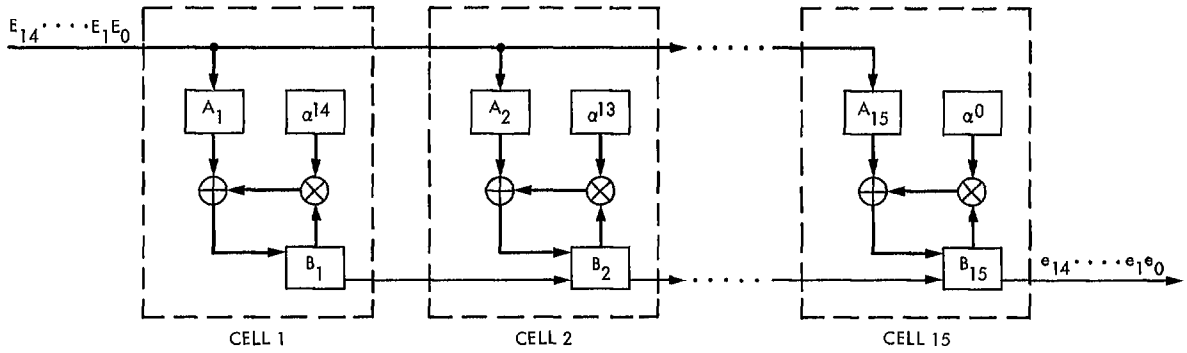

Fig. 8. The pipeline structure for computing error pattern in a (15, 11) RS code

Appendix A
The Construction of GF (2^4)

In this appendix, the construction of a finite field of 2^4 elements is given. To construct this field consider the irreducible polynomial $x^4 + x + 1$ over $GF(2)$. Let $\alpha \epsilon GF(2^4)$ be a root of $x^4 + x^2 + 1$, i.e., $\alpha^4 + \alpha^2 + 1 = 0$. The 15 nonzero field elements are given in Table A-1 in terms of root α.

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

Element	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