The Design Plan of a VLSI Single Chip (255, 223) Reed-Solomon Decoder

I. S. Hsu, H. M. Shao, and L. J. Deutsch
Communications Systems Research Section

The VLSI architecture of a single chip (255, 223) Reed-Solomon decoder for decoding both errors and erasures is described in this article. A decoding failure detection capability is also included in this system so that the decoder will recognize a failure to decode instead of introducing additional errors. This could happen whenever the received word contains too many errors and erasures for the code to correct. The number of transistors needed to implement this decoder is estimated at about 75,000 if the delay for received message is not included. This is in contrast to the older transform decoding algorithm which needs about 100,000 transistors. However, the transform decoder is simpler in architecture than the time decoder. It is therefore possible to implement a single chip (255, 223) Reed-Solomon decoder with today's VLSI technology. An implementation strategy for the decoder system is presented. This represents the first step in a plan to take advantage of advanced coding techniques to realize a 2.0-dB coding gain for future space missions.

I. Introduction

A concatenated coding system consisting of a convolutional inner code and a Reed-Solomon outer code has been adopted as a guideline for downlink telemetry for future space missions by CCSDS (Consultative Committee for Space Data Systems) (Ref. 1). The convolutional inner code is the same (7, 1/2) code used by NASA's Voyager project. The outer Reed-Solomon code is a (255, 223) block code of 8-bit symbols and it is capable of correcting t errors and s erasures if $2t + s \leqslant 32$. The performance of such schemes is investigated in Ref. 2 where it is shown that this concatenated channel provides a coding gain of almost 2 dB over the convolutional-only channel at a decoded bit error rate of 10^{-5} using only the error correcting capability of the codes. In Ref. 3, a (255, 223) Reed-Solomon encoder using Berlekamp's bit-serial multiplication algorithm is developed and proved to perform well. However, due to the sophisticated procedures involved in the Reed-Solomon decoding algorithm, especially the portion to perform the Euclid's algorithm, it is much more difficult to design a Reed-Solomon decoder in VLSI.

Recently, Brent and Kung (Ref. 4) suggested a systolic array architecture to compute the greatest common divisor (GCD) of two polynomials. By the use of this idea, a VLSI design of a pipeline Reed-Solomon decoder was developed (Ref. 6). Hsu et al. used this idea to implement a VLSI chip for calculating the GCD of two polynomials with 8-bit coefficients (Ref. 5). More recently, the pipeline design of the Reed-Solomon decoder was revised in Ref. 5. In this revision

the transform decoding algorithm is replaced by a time domain algorithm (see Appendix B). Erasure correction capability is included in the revision, and a multiplexed design for the Euclid's algorithm is used instead of the systolic array design. These improvements reduce the circuitry needed for VLSI implementation of Reed-Solomon decoder.

In this article, the VLSI architecture of the (255, 223) Reed-Solomon decoder for decoding both errors and erasures is described. The functional behavior of each block will be explained and the number of transistors needed for VLSI implementation is estimated. Finally, comparisons with the previous design are included as well.

This article represents a first cut at a final implementation plan for a Reed-Solomon decoding subsystem that may be used to support future deep space missions within the Deep Space Network (DSN). It also constitutes the first step toward the realization of a 2.0-dB coding gain through the use of advanced error correcting coding strategies.

II. The Decoding Procedure

Let $N = 2^m - 1$ be the length of a (N, I) Reed-Solomon code over $GF(2^m)$ with design distance d. Suppose that t errors and s erasures occur, and $s + 2t \leqslant d$. Then Reed-Solomon coding theory implies that the original codeword may be recovered from the received data.

First some definitions are needed. Let each X_i be an error location or an erasure location, and define the two sets $\Lambda = \{X_i | X_i$ is an erasure location$\}$, and $\lambda = \{X_i | X_i$ is an error location$\}$. Let Y_i be the corresponding errata magnitude and let $r = (r_0, r_1, \ldots, r_{N-1})$ be the received vector. Now the decoding process may be described in terms of the following basic steps.

Step 1: Compute the syndrome polynomial

$$S(Z) = \sum_{k=1}^{\infty} S_k Z^{-k}$$

where

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

The numbers S_k are called the "syndromes" of the received word.

Step 2: Compute the erasure locator polynomial

$$\Lambda(Z) = \prod_{X_i \epsilon \Lambda} (Z - X_i) \tag{2}$$

Step 3: Multiply $S(Z)$ and $\Lambda(Z)$ to obtain the Forney syndrome polynomial

$$T(Z) = S(Z)\,\Lambda(Z) \tag{3}$$

Step 4: Compute the errata evaluator polynomial $A(Z)$ and the error locator polynomial $\Lambda(Z)$ from

$$T(Z) = \frac{A(Z)}{\Lambda(Z)}$$

by a modified Euclid's algorithm.

Step 5: Multiply $\lambda(Z)$ and $\Lambda(Z)$ to get the errata locator polynomial

$$P(Z) = \lambda(Z)\,\Lambda(Z) \tag{4}$$

Step 6: Perform a Chien search on $P(Z)$ to find the error location set λ and the erasure location set Λ.

Step 7: Compute the errata magnitudes

$$Y_k = \frac{A(X_k)}{X_k P'(X_k)}, \qquad 1 \leqslant k \leqslant s + t$$

by evaluating $A(Z)$ and $P'(Z)$, the derivative of $P(Z)$. Use the sets Λ and λ to direct the addition of Y_k to the received vector r to produce the decoded result.

The extra calculation required to recognize a failure to decode was left out of the above discussion for clarity. It is explained in Ref. 7.

III. Functional Description and Transistor Estimations

In this section, the decoder is broken down into several basic blocks. The function of each block is described, and the number of transistors needed for VLSI implementation is estimated. The discussion here will be more detailed than in Ref. 7.

The estimates of the number of transistors required for the various blocks are based on the considerable work that has already been done on this project and the combined expertise

of a team of logic and VLSI designers within the Digital Signal Processing Research Group.

Figure 1 shows the overall architecture of the VLSI (255, 223) Reed-Solomon decoder. The decoder is divided into twelve blocks as described below.

(1) *Syndrome Transform*: This block calculates the syndromes from the received 255 symbol messages. The output of this block is the syndrome polynomial $S(Z)$ (see Fig. 1). Figure 4 shows a more detailed diagram of the syndrome transform block. The architecture of this block is similar to that of the existing Reed-Solomon encoder chips (Ref. 3) since one of the multiplicands is fixed. Therefore the Berlekamp multiplier is used here. This finite field multiplier design is simple and a large number of gates are saved as compared to other schemes (Ref. 8). The calculated syndromes are fed to the Polynomial Expansion I block in parallel as the design of polynomial expansion circuit needs to load the multiplicand polynomial all at once. A parallel to serial conversion circuit is therefore saved. The number of transistors needed is about

$$\underset{\text{Registers}}{32 \times 48} + \underset{\text{Multipliers}}{32 \times 16} + \underset{\text{XOR's}}{32 \times 10} = 2{,}300$$

where we assume one symbol register contains 48 transistors, 16 transistors are required in a basic cell of a dual basis multiplier, and there are 32 XOR's with each XOR containing 10 transistors (Ref. 8). The previous syndrome computation circuit (Ref. 6) needed about

$$32 \times 400 = 12{,}800$$

transistors, since 32 syndromes are calculated and each syndrome cell contains 400 transistors.

(2) *Power Calculation*: This block converts the input erasure data into a sequence of α^k's and 0's. Since the maximum erasure decoding capability of this decoder is 32, only 32 symbol latches are needed here. Figure 3 shows a block diagram of the power calculation block. The multiplier used is a standard basis multiplier with a fixed multiplicand α. This is because output of this block must be in the standard basis and because the circuitry for Berlekamp's multiplier with the added dual-to-standard basis conversion is more complex than that simply using the slightly more complex standard basis multiplier only. A detection circuit for detecting the occurrence of erasures is included. If an erasure in the k_ith location occurs, its corresponding symbol α^{k_i} is calculated and latched. A counter is used to count the number of erasures. If this number exceeds 32, a decoding failure alarm will result and the received message will be passed without decoding. The number of transistors needed in this block is about

$$\underset{\text{Registers}}{32 \times 48} + \underset{\text{Power Calculation}}{500} = 2{,}000$$

where we assume one symbol register contains 48 transistors and 500 transistors are needed in the power calculation circuitry.

(3) *Polynomial Expansion I*: This block performs polynomial multiplication of the syndrome polynomial $S(Z)$ and the erasure locator polynomial $\Lambda(Z)$ to obtain the Forney syndrome polynomial (see Fig. 1). Figure 5 shows a detailed block diagram of the polynomial expansion block. In Fig. 5, the multiplicand polynomial is entered in parallel while the multiplier polynomial comes in bit by bit. The output Forney syndrome polynomial is fed to the modified Euclid's algorithm stage serially as required in Ref. 5. Therefore a parallel to serial conversion circuit is included. The number of transistors needed in this block is about

$$32 \times 350 = 11{,}200$$

This is because the maximum degree of the Forney syndrome polynomial is 31 for this code. Therefore 32 subcells for 32 coefficients are needed. We assume 350 transistors are used in each subcell.

(4) *Delay I*: This block delays the erasure locator polynomial $\Lambda(Z)$ to synchronize it with the error locator polynomial $\Lambda(Z)$ which comes out of the modified Euclid's algorithm unit (see Fig. 1). This block consists of a series of shift registers. An external signal is used to control the latch operation, i.e., $\Lambda(Z)$ is latched for a certain amount of time and then released to the next stage when the error locator polynomial is ready. Figure 6 presents the block diagram of this part. Since erasure locations come into the chip continuously, 5-stage multiplexing (see Ref. 7) is anticipated in this design. The number of transistors needed is then about

$$5 \times 32 \times 8 \times 8 = 10{,}240$$

where we assume five 32-symbol latches are needed. A symbol latch contains 8 bits and each bit has 8 transistors.

(5) *Polynomial Expansion II*: This block performs the polynomial multiplication of erasure locator polynomial

$\Lambda(Z)$ and error locator polynomial $\lambda(Z)$ to obtain the errata polynomial $P(Z)$. The errata polynomial is fed to the next stages in parallel (see Fig. 1). This block is similar to Polynomial Expansion I except that the parallel to serial conversion circuit is not needed here. Figure 7 shows its block diagram. The number of transistors needed is about

$$32 \times 300 = 9{,}600$$

where we assume 300 transistors are contained in each subcell and there are 32 subcells.

(6) *Modified Euclid's Algorithm*: This block performs the modified Euclid's algorithm. It was calculated in Ref. 7 that only 5 GCD (greatest common divisor) subcells are needed instead of the 32 subcells required in Ref. 6 (see also Appendix A). This is because a multiplexing method is used. Figure 8 shows the block diagram of a multiplexed GCD cell. Since each GCD subcell contains about 4000 transistors (Ref. 5), this block contains about $5 \times 4000 = 20{,}000$ transistors. The outputs are the error locator polynomial $\lambda(Z)$ and the errata evaluator polynomial $A(Z)$ (see Fig. 1). The error locator polynomial is fed to the Polynomial Expansion II in parallel, while the errata evaluator polynomial is fed to Polynomial Evaluation bit by bit.

(7) *Polynomial Evaluation*: This block performs the evaluation of the errata evaluator polynomial $A(Z)$. Figure 9 shows the block diagram of this circuit. The polynomial $A(Z)$ is the input of this block; outputs are the evaluated values of the $A(X_k)$'s. Because the architecture of this circuit is similar to that of syndrome calculation, the number of transistors needed in this block is about

$$2{,}300 + 2{,}400 = 4{,}700$$

where 2400 transistors are used for implementing the summation operation. The $A(X_k)$'s are fed into the next stage serially.

(8) *Derivative, Evaluation, Multiplication, and Inverse (DEMI)*: This block takes the derivative of the polynomial $P(Z)$, and performs the evaluation, multiplication by X_k, and inverse of the final product. Figure 10 shows its block diagram. The derivative of the polynomial $P(Z)$ is calculated by merely dropping the coefficients of even terms since the field in which the operations take place is of characteristic 2 (Ref. 7). Evaluation of the polynomial $P'(Z)$ is similar to the Polynomial Evaluation block except that there are only 16 coefficients in $P'(Z)$ – only half that in the Polynomial Evaluation block. Hence about 2,350 transistors are needed for this polynomial evaluation. In total, the number of transistors needed in this block is about

$$2{,}350 + 2{,}500 = 4{,}850$$

where we assume 2,500 transistors are needed for other parts. The output is fed to the next stage serially.

(9) *Chien Search*: This block performs the Chien search algorithm for both the error and erasure locations. The outputs are the error and erasure locations. This circuit is similar to that of the Polynomial Evaluation block since the Chien search algorithm actually is a polynomial evaluation process (Ref. 10). Hence the number of transistors needed in this block is about 4,700. The estimated error and erasure locations are fed to the next stage serially.

(10) *Delay II*: This block is the delay for received messages which are used together with estimated errors and erasures to obtain the estimated infomation. Because the received messages are fed into the chip serially and continuously, a pipeline register array is needed. This means that the delay cannot be multiplexed in the same way as the GCD operation. The number of transistors needed is about

$$1{,}343 \times 48 = 65{,}000$$

where 1,343 symbol delays are estimated to be needed and one symbol delay contains about 48 transistors. This block occupies almost half of the area of the decoder.

(11) *Delay III*: This block is used to delay the $A(X_k)$'s for synchronization with the output from the DEMI block. These two are sent to a multiplier to form the errata magnitude (see Fig. 1). It is estimated that 16 symbol delays are required; hence the number of transistors needed in this block is

$$16 \times 48 = 768$$

where 48 transistors are needed in one symbol delay.

(12) *Decoding Failure Detection*: This circuit performs the decoding failure detection. The algorithm for the decoding failure test will not be described here. A description of this algorithm may be found in Ref. 9. Suppose the Hamming weight of the errata vector computed by the decoder is w and that t errors and s erasures have occurred. The decoder has erred if $2w >$

$d + s$, where d is the design distance of this Reed-Solomon code, which is 33. It is estimated that this circuit contains about 1,500 transistors.

The total number of transistors needed to implement the (255, 223) Reed-Solomon decoder, if we exclude Delay II, is about

$$2{,}300 + 2{,}300 + 11{,}200 + 10{,}240 + 9{,}600$$
$$+ 20{,}000 + 4{,}700 + 4{,}850 + 4{,}700 + 768$$
$$+ 1{,}500 = 72{,}000$$

If overheads are included, the maximum number of transistors should not exceed 75,000. However, if Delay II is included in the calculation, 75,000 + 65,000 = 140,000 transistors are needed for this decoder. This would greatly reduce the probability of a single chip implementation using the fabrication processes that are currently open to JPL. However, the Delay II block, while being the largest single block in the design, is also the simplest and the most easily implemented with off-the-shelf technology. The whole block can be realized with a single medium density random access memory (RAM) chip.

IV. Improvements Over the Previous Design

Basically, the VLSI architecture of this new (255, 223) Reed-Solomon decoder is similar to that in Ref. 7. Figure 2 presents the VLSI architecture of the previous Reed-Solomon decoder. Several important improvements have been achieved which substantially increase the performance of the system. The improvements include the following:

(1) The long delay line in the input stage for storing the erasure locations is now eliminated (see Fig. 1). This results from the fact that the Chien search procedure searches for both error and erasure locations while the previous design searches only for error locations (Ref. 7; see Fig. 2). This design alleviates the necessity of storing erasure locations. About 1,200 symbol delays are reduced by this revision. When represented in terms of transistor numbers, it is approximately

$$1{,}200 \times 8 \times 6 = 57{,}600$$

where we assume one symbol register contains 8 bits and each bit has 6 transistors.

(2) Two polynomial multiplication blocks used in Ref. 7 are replaced by two polynomial expansion circuits (see Fig. 1). By this revision, the polynomial expansion block in the input stage of erasure locations in Ref. 7 is eliminated. This means a reduction of about 12,800 transistors.

(3) The power calculation block (this block calculates powers of symbols according to erasure locations) which was not presented in Ref. 7 is refined here. Figure 3 shows its block diagram. The 255 symbol latches needed in the previous design are now reduced to 32. This is due to the fact that an erasure detection circuit is added to detect the occurrence of erasures. If an erasure occurs, its location will be latched and moved one symbol forward if the next erasure occurs; otherwise it remains latched. Since the erasure correcting capability of this code is 32, only 32 symbol latches are needed. If the number of erasures occurred is greater than 32, a decoding failure alarm will be given. This saves 223 symbol latches, i.e., 223 × 48 = 10,704 transistors since each symbol latch contains 48 transistors.

(4) Berlekamp's multiplication algorithm (Ref. 8) is used in this decoder except in the power calculation block. It was discovered in Ref. 8 that the dual basis multiplication algorithm is the simplest among all known finite field multipliers. Also, the conversion between dual basis and standard basis is not complicated. Therefore if basis conversion is not used too often, the dual basis multiplier is the best choice. This is indeed the case in the Reed-Solomon decoder design.

Although the normal basis multiplier was used in the GCD design (Ref. 5), the revision is simple. The only modification is to replace the normal basis multiplier by the dual basis multiplier. The remainder of the circuit will be left unchanged. By this revision, basis conversions from normal to standard basis and dual basis are totally eliminated.

Due to the simplicity of Berlekamp's multiplication algorithm, the number of gates used in this design represents a substantial reduction over the previous ones. It is estimated that a Berlekamp general purpose multiplier needs about 400 transistors, while the Massey-Omura general purpose multiplier needs about 500 transistors. Therefore, if the multipliers are used frequently, the savings in the number of transistors is tremendous. In blocks such as Syndrome Transform, Polynomial Evaluation, and Chien Search where one of the multiplicands is fixed, the advantage of using the dual basis multiplier is more predominant.

Since there is only one multiplier in the power calculation circuit, the output must be in standard basis for compatibility

with next stage. For this reason, a standard basis multiplier is used in this block.

V. Implementation Procedure

The first step in the implementation of this decoder has already been accomplished. The various algorithms and architectures have been tested through analysis, simulation, and, in many cases, actual VLSI implementation. The next step is to implement versions of the various blocks so as to develop a complete Reed-Solomon decoding system. We plan to build the blocks described above on several separate chips. This will enhance their testability. When they are fully tested, the next step will be to integrate these working chips into 4 subchips as categorized in the following:

CHIP 1: This chip contains Delay II. The number of transistors needed is about 65,000. This chip may very well end up being an off-the-shelf RAM chip.

CHIP 2: This chip contains Syndrome Transform, Power Calculation, Polynomial Expansion I, and Delay I. The number of transistors is about

$$2{,}300 + 2{,}900 + 9{,}600 + 10{,}240 = 24{,}000$$

CHIP 3: This chip performs the modified Euclid's algorithm and contains about 20,000 transistors.

CHIP 4: This chip contains the following circuits: Polynomial Expansion II, Polynomial Evaluation, Chien Search, DEMI, Delay III, and Decoding Failure Detection. The number of transistors needed is about

$$9{,}600 + 4{,}400 + 4{,}400 + 4{,}700 + 768 + 1{,}500 = 25{,}400$$

When all of the above four chips are proven to work together as a system, the final step will be to put the whole decoder on a single silicon chip. It will be the first single VLSI chip (255, 223) Reed-Solomon decoder.

References

1. "Recommendation for Space Data System Standards: Telemetry Channel Coding," (Blue Book), Consulatative Committee for Space Data System, NASA, May 1984.
2. Miller, R. L., Deutsch, L. J., and Butman, S. A., "On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes," *JPL Publication 81-3*, Jet Propulsion Laboratory, Pasadena, Calif., Sept. 1981.
3. Hsu, I. S., Reed, I. S, Truong, T. K., Wang, K., Yeh, C. S., and Deutsch, L. J., "The VLSI Implementation of a Reed-Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm," *IEEE Trans. on Computers*, Vol. c-33, No. 10, pp. 906–911, Oct. 1984.
4. Brent, R. P., and Kung, H. T., "Systolic VLSI Arrays for Polynomial GCD Computations," Dept. Computer Science, Carnegie-Melon University, Pittsburgh, PA, Rep., 1982.
5. Hsu, I. S., Deutsch, L. J., Shao, H. M., and Truong, T. K., "A Single VLSI Chip for Polynomial GCD Computation," to be published in *TDA Progress Report*, Jet Propulsion Laboratory, Pasadena, Calif.
6. Shao, H. M., Truong, T. K., Deutsch, L. J., Yuen, J. H., and Reed, I. S., "A VLSI Design of a Pipeline Reed-Solomon Decoder," *IEEE Trans. on Computers*, Vol. c-34, No. 5, pp. 393–403, May 1985.
7. Shao, H. M., Truong, T. K., Hsu, I. S., Deutsch, L. J., and Reed, I. S., "A Single Chip VLSI Reed-Solomon Decoder," submitted to *IEEE Trans. on Computers*.

8. Hsu, I. S., Truong, T. K., Shao, H. M., Deutsch, L. J., and Reed, I. S., "A Comparison of VLSI Architecture of Finite Field Multiplier Using Dual, Normal, and Standard Basis," submitted to *IEEE Trans. on Computers*.

9. Miller, R. L., Truong, T. K., and Reed, I. S., "A Decoding Failure Test for the Transform Decoder of Reed-Solomon Codes," *TDA Progress Report 42-62*, Jet Propulsion Laboratory, Pasadena, Calif., pp. 121–124, Jan. 1981.

10. Peterson, W. W., and Weldon, E. L., *Error Correcting Codes*. Cambridge, Massachusetts: MIT press, 1980.

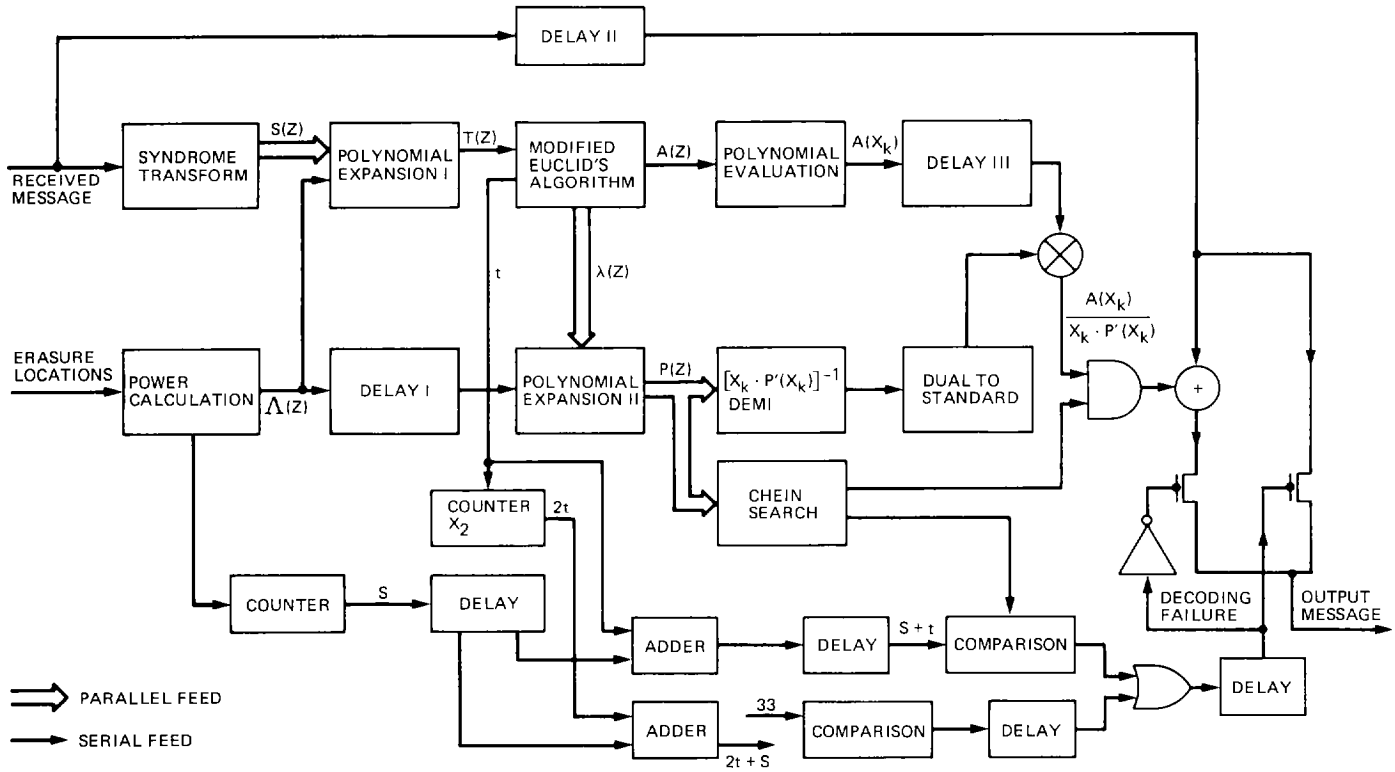

Fig. 1. Block diagram of the (255, 223) Reed-Solomon decoder for decoding both errors and erasures with decoding failure detection

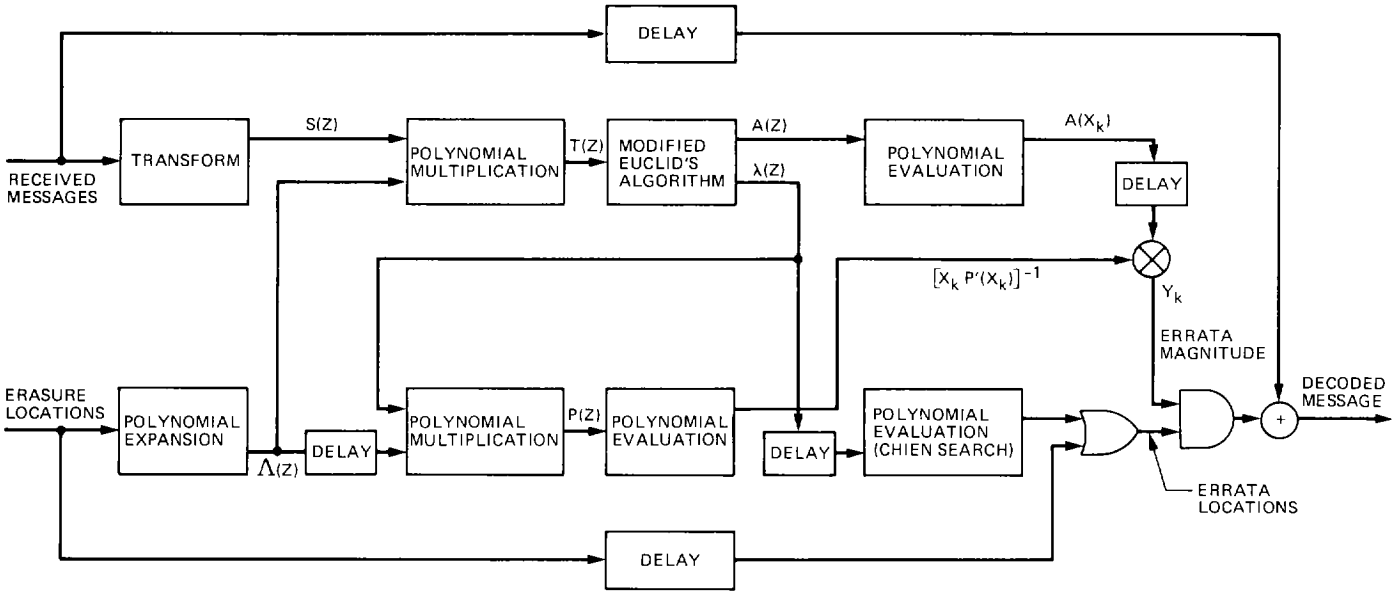

Fig. 2. VLSI architecture of the previously designed pipelined Reed-Solomon decoder for both error and erasure corrections

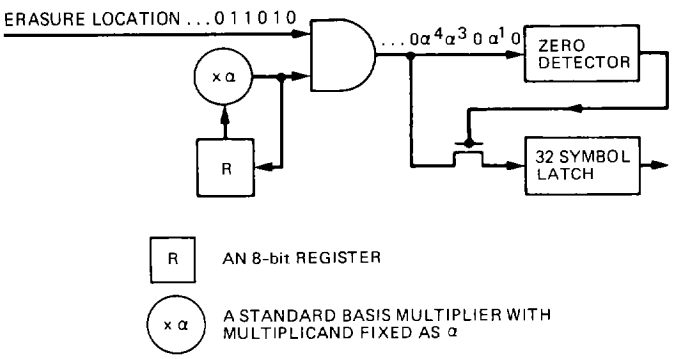

Fig. 3. The block diagram of the Power Calculation part

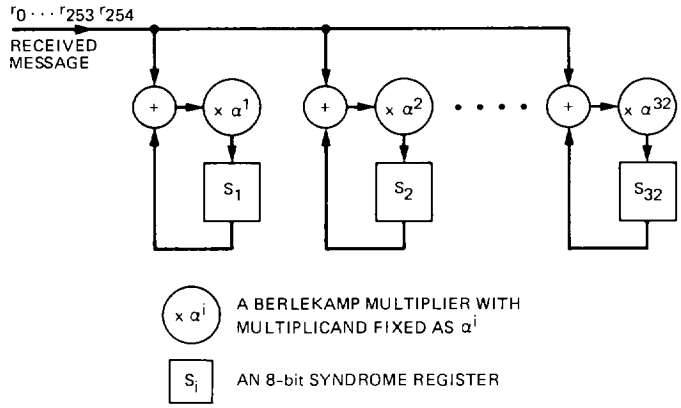

Fig. 4. The logic diagram of a Syndrome Transform block

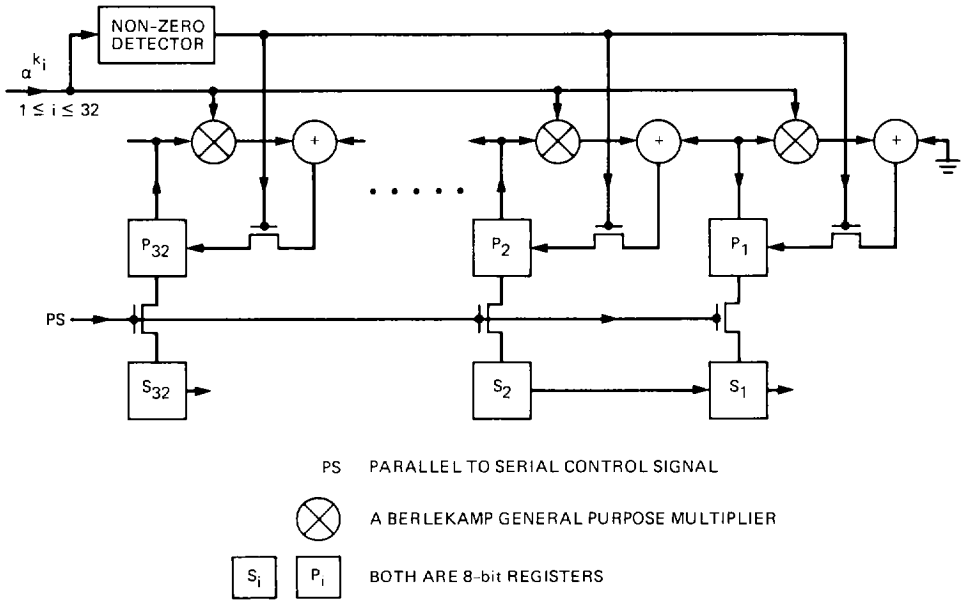

Fig. 5. The logic diagram of Polynomial Expansion I

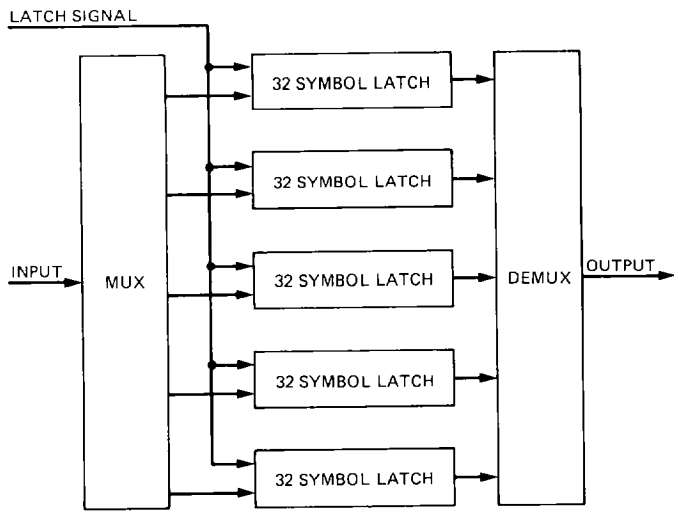

Fig. 6. The block diagram of Delay I

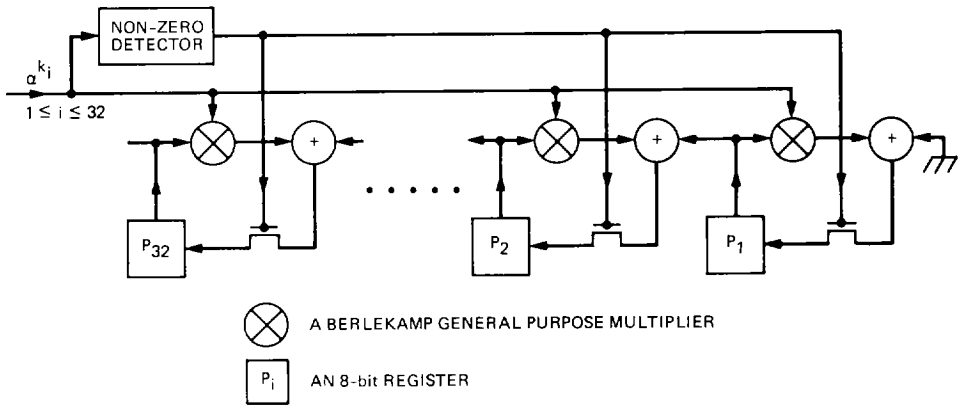

Fig. 7. The logic diagram of Polynomial Expansion II

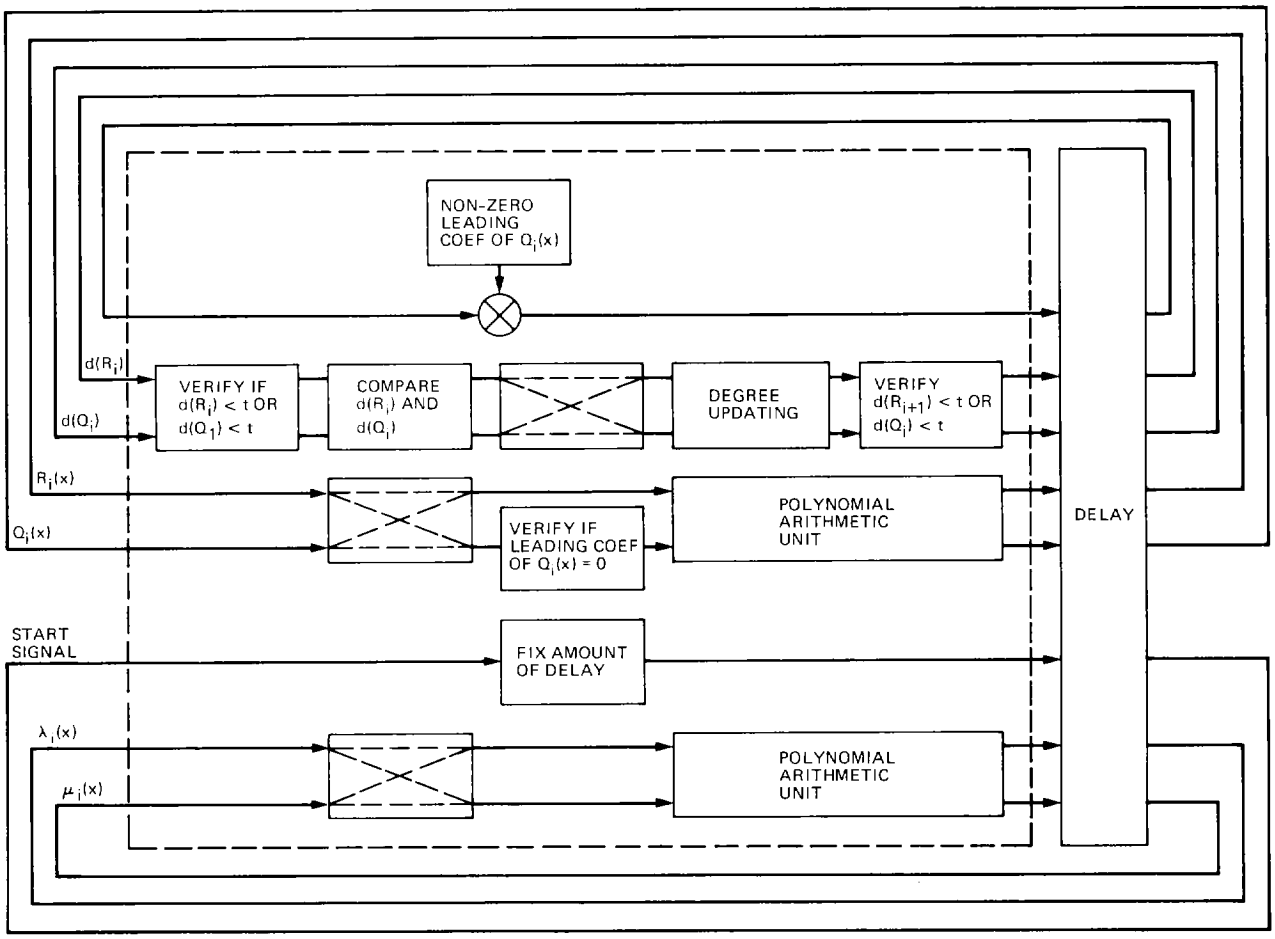

Fig. 8. The block diagram of a multiplexed GCD subcell

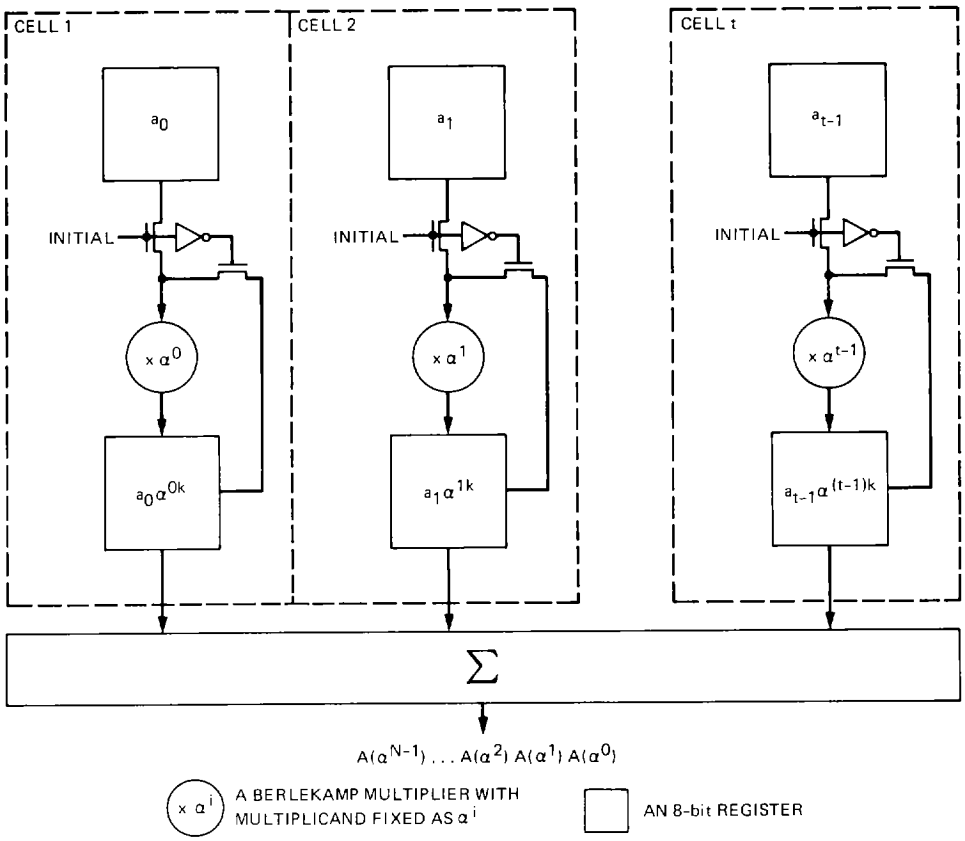

Fig. 9. The block diagram of the Polynomial Evaluation part

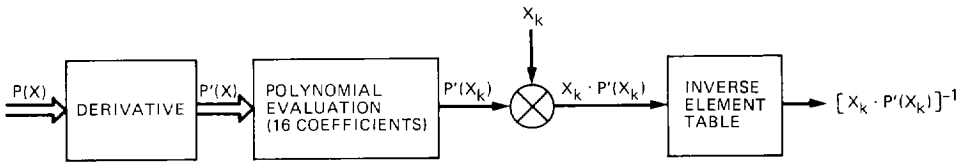

Fig. 10. The block diagram of DEMI

Appendix A

Comparison of the New Euclid's Algorithm With the Old

In this appendix, a comparison between the previous fully pipelined Reed-Solomon decoder design and the new multiplexed Reed-Solomon decoder design as described in this article is exhibited. The previous pipeline design of GCD cells needs about

$$\underset{\text{GCD Subcells}}{32 \times 4{,}000} + \underset{\text{Delays}}{160 \times 48} = 135{,}680$$

transistors, since 32 GCD subcells and 160 symbol delays are needed in the modified Euclid's algorithm block (Ref. 6). Figure A-1 shows the block diagram of the multiplexed modified Euclid's algorithm. For this new Euclid's algorithm, the number of transistors needed is about

$$\underset{\text{GCD Subcells}}{5 \times 4{,}000} + \underset{\text{Delays}}{5 \times 32 \times 8 \times 8} = 30{,}240$$

since only 5 stages of GCD subcells and five 32-symbol delays are needed. Apparently, the new multiplexed GCD design is much simpler than the previous pipelined GCD design.

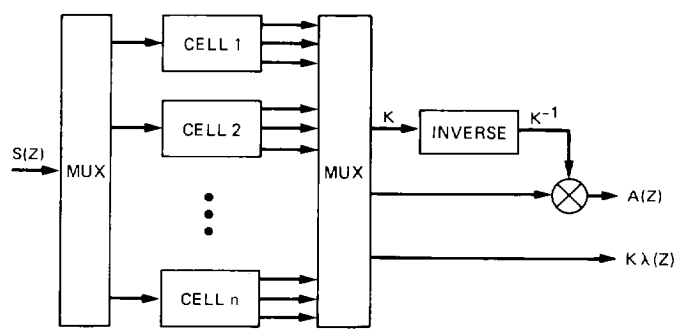

Fig. A-1. The block diagram of the multiplexed modified Euclid's algorithm

Appendix B

Transistor Count for the Transform Design

In this appendix, the number of transistors needed for using the transform algorithm to design a Reed-Solomon decoder is estimated. It will show that the transform decoding algorithm uses more transistors than the new algorithm as described in this article.

The VLSI architecture using transform methods to decode the Reed-Solomon code is quite different from that described in this article. Figure B-1 exhibits the VLSI architecture of such a Reed-Solomon decoder. Blocks such as Polynomial Evaluation, DEMI, Chien Search, and Delay III which are used in the present design are not needed. However, circuits for calculating Extended Syndrome from the coefficients of the combined error and erasure locator polynomial as well as syndrome, and inverse transform of error patterns are included. Also, the syndrome delay is necessary in this design for the reason stated above. It is estimated that this delay needs 6 stages of multiplexing. Hence the number of transistors needed for this part is about 6 X 32 X 8 X 8 = 12,300.

Since 32 cells are needed in the transformed error pattern calculation, and each cell needs about 400 transistors, 32 X 400 = 12,800 transistors are needed to implement this part. The architecture for the inverse transform of error patterns is similar to that of calculating syndrome except that 255 cells are needed instead of 32. That is to say, the inverse transform circuit is approximately 8 times larger than the syndrome calculation circuit. Hence the number of transistors needed for the inverse transform is about 8 X 2,300 = 18,400.

The total number of transistors needed for implementing the (255, 223) Reed-Solomon decoder using the transform decoder algorithm is about

$$2{,}300 + 2{,}300 + 9{,}600 + 10{,}240$$

$$+ 20{,}000 + 1{,}500 + 9{,}600 + 12{,}300$$

$$+ 12{,}800 + 18{,}400 = 98{,}440$$

If overhead is included, the number should not exceed 10,000 while the present design needs only 75,000 transistors. Note that the input message delay, Delay II, is not counted in this estimation.

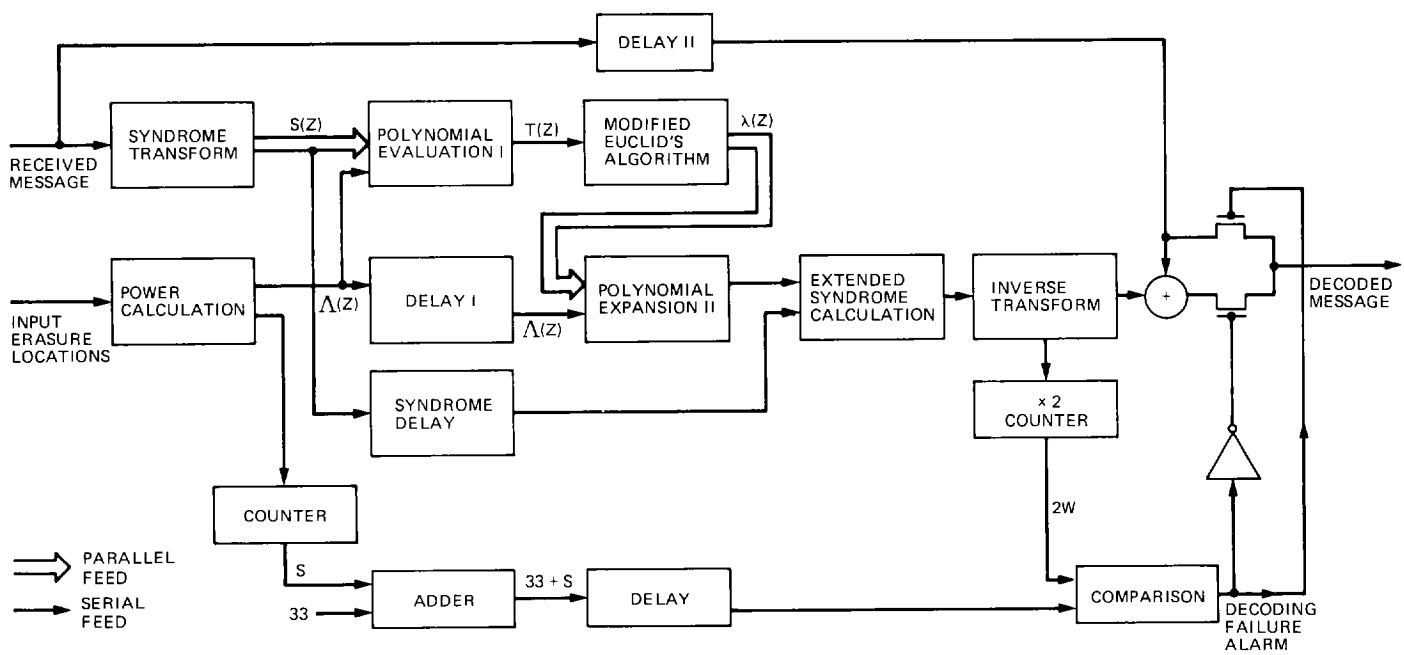

Fig. B-1. The VLSI architecture of a Reed-Solomon decoder using the transform decoding algorithm