A VERY LOW-COST MULTI-MODE REED SOLOMON DECODER BASED ON PETERSON-GORENSTEIN-ZIERLER ALGORITHM

Sheng-Feng Wang
Department of Electrical Engineering,
National Taiwan University
Taipei 106, TAIWAN, R.O.C.

Huai-Yi Hsu and An-Yeu Wu
Institute of Electronics Engineering,
National Taiwan University
Taipei 106, TAIWAN, R.O.C.

Abstract *Reed-Solomon* (*RS*) codes play an important role in providing error protection and data integrity. Among various RS decoding algorithms, the *Peterson-Gorenstein-Zierler* (*PGZ*) in general has the least computational complexity for small t values. However, unlike the iterative approaches (*e.g.*, *Berlekamp-Massey* algorithm), it will encounter divided-by-zero problems in solving multiple t values. In this paper, we propose a multi-mode hardware architecture for error number ranging from zero to three. We first propose a cost-down techniques to reduce the hardware complexity of a $t=3$ decoder. Then, we perform algorithmic-level derivation to identify the configurable feature of our design. With the manipulations, we are able to perform multi-mode RS decoding in one unified VLSI architecture with very simple control scheme. The very low cost and simple datapath make our design a good choice in small-footprint embedded VLSI systems such as *Error Control Coding* (*ECC*) in memory systems

INTRODUCTION

Reed-Solomon (*RS*) code has a widespread use for forward error correcting in digital transmission and storage systems. It is a special case of BCH codes, and has become a popular choice to provide data integrity due to its good error correction capability for burst transmission errors [1][2][3].

Among various RS decoding algorithms, the *Peterson-Gorenstein-Zierler* (*PGZ*) algorithm [4][5] provides the simplest way to realize the RS decoder for $t \leq 3$. It is very cost-effective for systems that require only small correcting capability, *e.g.*, *Error Control Coding* (*ECC*) in processor-memory systems and digital answer machines. Unlike the iterative RS decoding methods (*e.g.*, *Berlekamp-Massey* algorithm [6][7]), the major drawback of the conventional PGZ algorithm works for only single correction capability. That is, the PGZ circuit to solve $t=3$ cannot function correctly if t is *1* or *2*. As a result, a $t \leq 3$ PGZ decoder will need three copies of hardware components to compute $t=1$, $t=2$, and $t=3$, respectively. The whole circuit is shown in Figure 1 (a).

0-7803-7145-3/01/$10.00 ©2001 IEEE

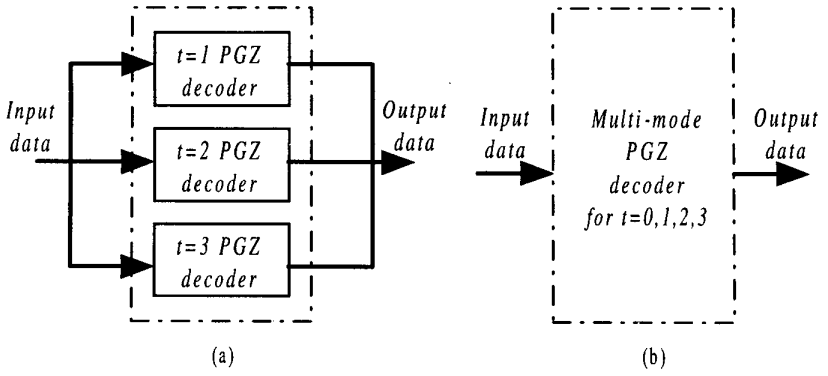

Figure 1. (a) Three copies of PGZ decoders based on conventional design approach (b) The proposed multi-mode PGZ decoder

Obviously, placing three copies of decoders on a circuit will definitely be a waste of silicon area and cost. We seek a simple way to merge three different decoders into one unified VLSI circuit, In this paper, we derive a configurable VLSI architecture to perform the multi-mode RS code for various correction capabilities (*i.e.*, different t values) based on the *Peterson-Gorenstein-Zierler* (*PGZ*) algorithm. We call it Multi-mode PGZ decoder as illustrated in Figure 1 (b). The reconfigurable feature of the proposed multi-mode PGZ decoder can solve $t=0,1,2,3$ errors altogether, which leads to significant saving in hardware cost.

The rest of this paper is organized as follows. In Sec. 2, we go through the details of the PGZ decoding algorithm. Then, we derive the reduced-complexity RS decoder for $t=3$. In Sec. 3 and 4, we present the multi-mode RS decoder. In Sec 5, we discuss the hardware complexity to illustrate the hardware saving of our approach. Finally, we conclude our work in Sec. 6.

REVIEW OF PGZ ALGORITHM

Syndrome Calculation

Let polynomial $c(x)$ denote the transmitted code word. Then the received code word, $r(x)$, can be represented as

$$r(x) = c(x) + e(x), \tag{1}$$

where $e(x)$ represents the error pattern. The syndrome values, denoted by S_i, are obtained by evaluating the received polynomial $r(x)$ at α_i. That is, equation can be written as

$$S_i = r(\alpha^i) = \sum_{j=0}^{n-1} r_j (\alpha^i)^j \quad , \quad 1 \le i \le 2t . \tag{2}$$

We also define *Syndrome polynomial* as

$$S(x) = \sum_{i=0}^{2t-1} S_{i+1} x.^{i} \tag{3}$$

PGZ algorithm

The PGZ algorithm includes two main steps. Solving *Newton Identity* is the first step:

$$\begin{bmatrix} S_2 & S_3 & \cdots & S_{t+1} \\ S_3 & S_4 & \cdots & S_{t+2} \\ \vdots & \vdots & \ddots & \vdots \\ S_{t+1} & S_{t+2} & \cdots & S_{2t} \end{bmatrix} \begin{bmatrix} \sigma_{t-1} \\ \sigma_{t-2} \\ \vdots \\ \sigma_0 \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_2 \\ \vdots \\ -S_t \end{bmatrix}. \tag{4}$$

That is, the syndrome values are used to solve for σ values in Eq. (4). Define the *Error location polynomial* as

$$\sigma(x) = \sigma_0 + \sigma_1 x + ... + \sigma_{t-1} x^{t-1} + x^t. \tag{5}$$

Then, we can solve the *Key equation*

$$\sigma(x) S(x) = -\omega(x) + \mu \cdot x^{2t}, \tag{6}$$

where the *Error value polynomial* is defined as

$$\omega(x) = \omega_0 + \omega_1 x + ... \omega_{t-1} x^{t-1}. \tag{7}$$

PGZ Algorithm for t =1

Given $t=1$, from Eq. (4), we have

$$[S_2][\sigma_0] = [-S_1], \text{ and } \sigma_0 = \frac{S_1}{S_2}. \tag{8}$$

Then we can *compute* the error location as

$$\sigma(x) = \sigma_0 + x. \tag{9}$$

Next, we *can* solve the key equation for $t=1$

$$\sigma(x) S(x) = -\omega(x) + \mu \cdot x^2, \tag{10}$$

$$\omega(x) = -(\sigma_0 + x)(S_1 + S_2 x) \bmod x^2, \tag{11}$$

where the *error* value polynomial is

$$\omega(x) = \omega_0, \text{ and } \omega_0 = \sigma_0 S_1. \tag{12}$$

PGZ Algorithm for *t*=2

For *t*=2, Eq. (4) is reduced to

$$\begin{bmatrix} S_2 & S_3 \\ S_3 & S_4 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_0 \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_2 \end{bmatrix}. \tag{13}$$

Then, we have

$$\sigma_0 = \frac{S_1 S_3 + (S_2)^2}{S_2 S_4 + (S_3)^2}, \quad \sigma_1 = \frac{S_2 S_3 + S_1 S_4}{S_2 S_4 + (S_3)^2}. \tag{14}$$

Then the error location polynomial can be written as

$$\sigma(x) = \sigma_0 + \sigma_1 x + x^2. \tag{15}$$

Solving the key equation for *t*=2 yields

$$\sigma(x) S(x) = -\omega(x) + \mu \cdot x^4, \tag{16}$$

$$\omega(x) = -(\sigma_0 + \sigma_1 x + x^2)(S_1 + S_2 x + S_3 x^2 + S_4 x^3) \bmod x^4, \tag{17}$$

Then, the error value polynomial can be represented as

$$\omega(x) = \omega_0 + \omega_1 x \tag{18}$$

with $\omega_0 = \sigma_0 S_1$ and $\omega_1 = \sigma_0 S_2 + \sigma_1 S_1$ (19)

PGZ Algorithm for *t*=3

Similarly, for *t*=3, we have

$$\begin{bmatrix} S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \\ S_4 & S_5 & S_6 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \\ \sigma_0 \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_2 \\ -S_3 \end{bmatrix}. \tag{20}$$

The coefficients of the error location polynomial can be solved as

$$\sigma_0 = \frac{S_2S_3S_4 + S_2S_3S_4 + S_1S_3S_5 + S_1S_4S_4 + S_2S_2S_5 + S_3S_3S_3}{S_2S_4S_6 + S_3S_4S_5 + S_3S_4S_5 + S_4S_4S_4 + S_3S_3S_6 + S_2S_5S_5},$$

$$\sigma_1 = \frac{S_2S_2S_6 + S_1S_4S_5 + S_3S_3S_4 + S_2S_4S_4 + S_1S_3S_6 + S_2S_3S_5}{S_2S_4S_6 + S_3S_4S_5 + S_3S_4S_5 + S_4S_4S_4 + S_3S_3S_6 + S_2S_5S_5}, \quad (21)$$

$$\sigma_2 = \frac{S_1S_4S_6 + S_2S_4S_5 + S_3S_3S_5 + S_1S_5S_5 + S_2S_3S_6 + S_3S_4S_4}{S_2S_4S_6 + S_3S_4S_5 + S_3S_4S_5 + S_4S_4S_4 + S_3S_3S_6 + S_2S_5S_5}.$$

Then, the error location polynomial can be written as

$$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + x^3. \quad (22)$$

The key equation for $t=3$ can be written as

$$\sigma(x)S(x) = -\omega(x) + \mu \cdot x^6 \quad (23)$$

and

$$\omega(x) = -\left(\sigma_0 + \sigma_1 x + \sigma_2 x^2 + x^3\right)\left(S_1 + S_2 x + S_3 x^2 + S_4 x^3 + S_5 x^4 + S_6 x^5\right) \bmod x^6, \quad (24)$$

where the error value polynomial is

$$\omega(x) = \omega_0 + \omega_1 x + \omega_2 x^2 \quad (25)$$

with $\omega_0 = \sigma_0 S_1$, $\omega_1 = \sigma_0 S_2 + \sigma_1 S_1$, $\omega_2 = \sigma_0 S_3 + \sigma_1 S_2 + \sigma_2 S_1$. (26)

Obviously, Eq. (21) turns out to be very complicated compared with Eqs. (8) and (14). The direct implementation of Eq. (21) will be tedious and complicated. Hence, in what follows, we provide a method to calculate to σ_0, σ_1, and σ_2 in a cost-efficient way.

The Reduced-Complexity Decoder Architecture for $t=3$

According to our observation, in Eq. (21) the denominator have two $S_3S_4S_5$ terms, which can be cancelled out on Finite-field addition. This condition can be applied to the numerator of σ_0, which contains two $S_2S_3S_4$ terms. We also discover that the term, S_2S_5, appears quite often in Eq. (21), *e.g.*, it is the common term of $S_2S_2S_5$, $S_2S_3S_5$, $S_2S_4S_5$, $S_2S_5S_5$. Thus, if we calculate S_2S_5 first, the overall computation complexity can be reduced significantly. Similarly, we can identify other common terms, such as S_2S_6, S_4S_4, S_3S_3, S_2S_5, S_1S_5, S_1S_6, and calculate them first, which leads to cost-efficient architecture as

illustrated in Figure 2. When σ_0, σ_1, and σ_2 are available, ω_0, ω_1, and ω_2 can be obtained from Eq. (26), as illustrated in Figure 3.

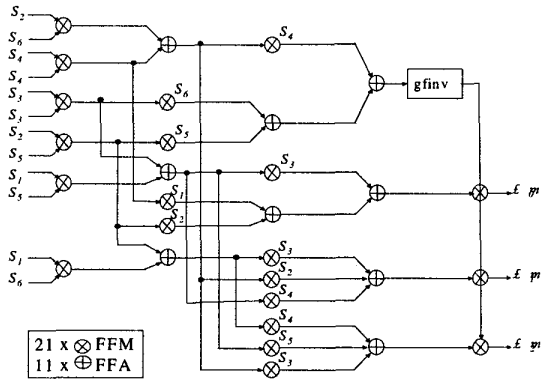

Figure 2. The block diagram of the t=3 PGZ architecture (σ part).

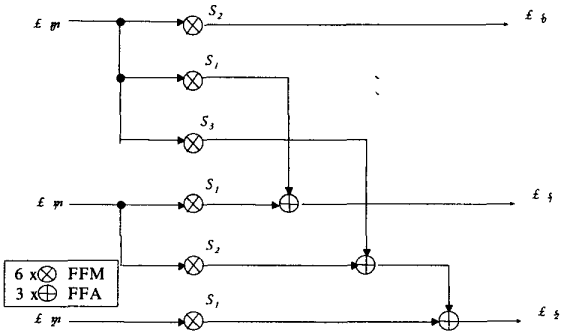

Figure 3. The block diagram of the t=3 PGZ architecture (ω part).

MULTI-MODE PGZ ALGORITHM AND ARCHITECTURE

Problems of t=3 PGZ Architecture when t=1 or 2

The block diagram introduced in Sec 2.D can function correctly only when the received code word has exactly three errors. However, if the error number is less than three, divided-by-zero problem will occur. Specifically, for $t=3$, we have to solve

$$\begin{bmatrix} S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \\ S_4 & S_5 & S_6 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \\ \sigma_0 \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_2 \\ -S_3 \end{bmatrix}. \tag{27}$$

If the error number is less than 3, the three columns of the 3-by-3 matrix will become linearly dependent, that is

$$\begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = \alpha \begin{bmatrix} S_3 \\ S_4 \\ S_5 \end{bmatrix} = \beta \begin{bmatrix} S_4 \\ S_5 \\ S_6 \end{bmatrix}, \text{ where } \alpha \text{ and } \beta \text{ are constants.} \quad (28)$$

Consequently, the denominator term and three numerator terms of the Eq. (21) are all equal to zero.

$$\begin{aligned} &S_2S_4S_6 + S_4S_4S_4 + S_3S_3S_6 + S_2S_5S_5 = 0 \\ &S_1S_3S_5 + S_1S_4S_4 + S_2S_2S_5 + S_3S_3S_3 = 0 \\ &S_2S_2S_6 + S_1S_4S_5 + S_3S_3S_4 + S_2S_4S_4 + S_1S_3S_6 + S_2S_3S_5 = 0 \\ &S_1S_4S_6 + S_2S_4S_5 + S_3S_3S_5 + S_1S_5S_5 + S_3S_3S_6 + S_3S_4S_4 = 0 \end{aligned} \quad (29)$$

Similarly, the denominator term and two numerator terms of σ's in Eq. (14) also become zero as long as the error number is less than 2.

$$\begin{aligned} &S_2S_4 + S_3S_3 = 0 \\ &S_1S_3 + S_2S_2 = 0 \\ &S_1S_4 + S_2S_3 = 0 \end{aligned} \quad (30)$$

Apparently, the σ values are now equal to divided-by-zero numbers, which cannot be manipulated anymore. Hence, the *t=3* architecture above cannot guarantee the right result given that *t=1* or *2*. To overcome this situation, three copies of hardware (Figure 1(a)) are needed, together with a specific state machine to check the error status.

The Proposed Multi-mode Decoding Algorithm

In fact, the zero values contain some information to facilitate our derivations. That is, by recognizing one of four terms in Eq. (29) and one of three terms in Eq. (30) the error number can be decided. For instance, $(S_2S_4S_6+S_4S_4S_4+S_3S_3S_6+S_2S_5S_5)$ will equal to zero when *t=0,1,2*; $(S_2S_4 + S_3S_3)$ will equal to zero when *t=0,1* and S_2 will equal to zero when *t=0*. Consequently, we employ these three terms to detect the error number *t*. Figure 4 shows the flowchart to detect the error number.

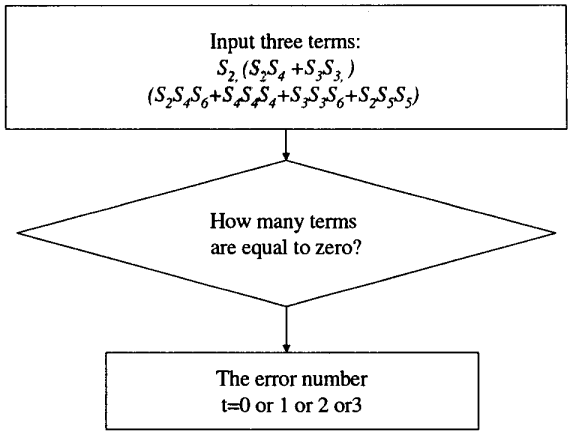

Figure 4. The flowchart to detect the error number in the proposed RS decoder.

By examining Eq. (14) carefully, we can discover that S_1S_3, S_2S_2, S_2S_4, S_3S_3, S_2S_3, S_1S_4, and $S_1S_3+S_2S_2$, $S_2S_4+S_3S_3$, $S_2S_3+S_1S_4$ are generated when calculating σ for $t=2$. Our approach is to compute σ for $t=3$ using these terms as basis. Meanwhile, as we mention in Sec 2.D, two $S_3S_4S_5$ terms and two $S_2S_3S_4$ terms can be neglected, which helps a lot in reducing the overall complexity. Although there are more hardware, the multi-mode PGZ decoder will generate the term needed to calculate different σ for $t=1,2,3$ at the same time. Providing that we know the error number, the correct term to calculate σ value can be chosen. Multiplexors in the multi-mode decoder will perform this selection. Figure 5 and Figure 6 show the block diagram of the proposed multi-mode PGZ decoder architecture. The algorithm of the controller will base on the flowchart in Figure 4.

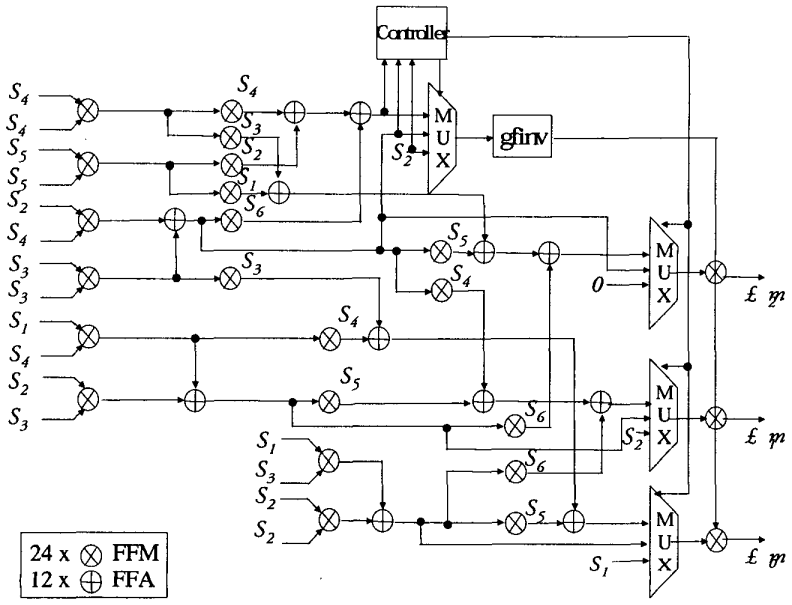

Figure 5. The block diagram of the multi-mode PGZ architecture (σ part).

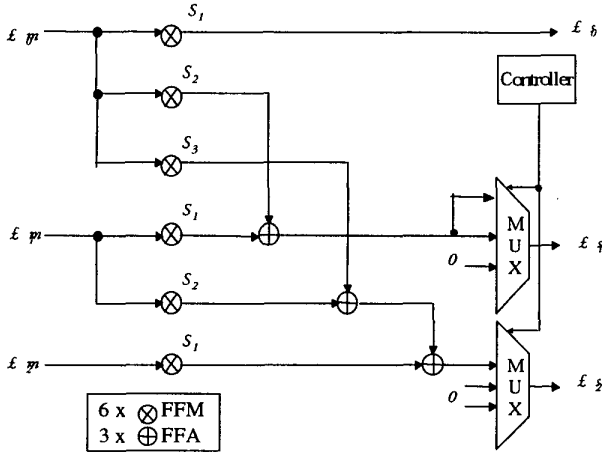

Figure 6. The block diagram of the multi-mode PGZ architecture (ω part).

MULTI-MODE CHIEN'S SEARCH & FORNEY'S METHOD

After locating all σ and ω values, the error location polynomial of Eq. (5) and the error value polynomial of Eq. (7) can be formed. According to Chien's search, the error location l satisfies the equation below.

$$\sigma(\alpha^{-l}) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_t x^t = 0, \quad \sigma_t = 1, \quad 0 \le l \le 2^m - 1. \tag{31}$$

where l denotes the error location. The error location polynomial reduces to Eqs. (32), (33), and (34), for $t=1$, $t=2$, and $t=3$, respectively:

$$\sigma(x) = \sigma_0 + 1x^1 + 0x^2 + 0x^3, \quad t=1, \tag{32}$$

$$\sigma(x) = \sigma_0 + \sigma_1 x^1 + 1x^2 + 0x^3, \quad t=2, \tag{33}$$

$$\sigma(x) = \sigma_0 + \sigma_1 x^1 + \sigma_2 x^2 + 1x^3, \quad t=3. \tag{34}$$

Suppose that we build a circuit to solve the equation for $t=3$ case, deliberately setting σ_2 to *1* for $t=2$ case, and σ_2 to *0*, σ_1 to *1* for $t=1$ case, the roots of Eqs. (32) (33) (34) can be searched, no matter what t is. The outcome from the multiplexor of the multi-mode decoder will pick up appropriate σ values.

Meanwhile, Forney's method is applied to find error value E_l, which corresponds to the error location l. Then we have

$$E_l = \frac{\omega(\alpha^{-\ell})}{\sigma'(\alpha^{-\ell})} = \frac{\alpha^{-\ell}\omega(\alpha^{-\ell})}{\sigma_{odd}(\alpha^{-\ell})} \tag{35}$$

where

$$\begin{aligned} \sigma'(x) &= \sigma_1 + 2\sigma_2 x + 3\sigma_3 x^2 \cdots + t\sigma_t x^{t-1} \\ &= \sigma_1 + \sigma_3 x^2 + \sigma_5 x^4 + \cdots = \frac{\sigma_{odd}(x)}{x} \end{aligned} \tag{36}$$

Setting t=1, 2 or 3, error value equations for three special cases can be expressed as

$$E_l = \frac{\omega_0 + 0x^1 + 0x^2}{1 + 0x^2}, \quad t=1, \tag{37}$$

$$E_l = \frac{\omega_0 + \omega_1 x^1 + 0x^2}{\sigma_1 + 0x^2}, \quad t=2, \tag{38}$$

$$E_l = \frac{\omega_0 + \omega_1 x^1 + \omega_2 x^2}{\sigma_1 + 1x^2}, \quad t=3. \tag{39}$$

The equation for $t=3$ case is obviously the most complicated. Therefore, once an architecture can resolve it, the equation for $t=2$ as well as $t=1$ can be calculated only by changing coefficients. The controller will control the multiplexor to select the proper ω values in Figure 6.

Figure 7 shows the implementation of Chien's search & Forney's method. The offset is the corrupted data, which must be added to corresponding error value E_l to produce the corrected data.

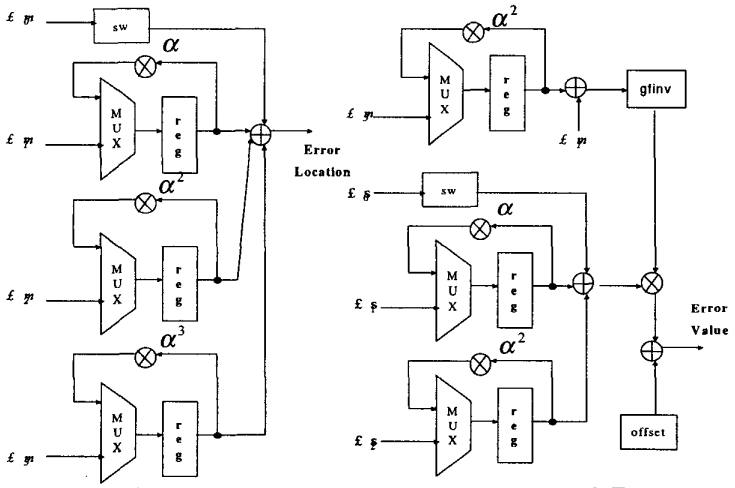

Figure 7. The block diagram of the proposed Chien's search & Forney's method.

COMPARISON OF COMPLEXITY

The main drawback of PGZ algorithm is that its hardware complexity will rise rapidly provided that t is larger than three. Direct implementation of the PGZ algorithm for $t=3$ without employing any cost-down techniques requires 40 Finite-field multiplier (FFM) and 16 Finite-field adder (FFA). By exploiting the special properties of the finite field operations in Sec 2.D, we had derived a reduced-complexity PGZ decoder for $t=3$. It requires only 21 FFM and 11 FFA and the hardware complexity is saved by approximately 50%. Furthermore, the design techniques of the reduced-complexity $t=3$ design is applied to our multi-mode PGZ decoder for any $t \leq 3$. It needs only 24 FFM and 12 FFA. The comparison of hardware complexity is shown in Table 1. As we can see, compared with the reduced-complexity designs, only three additional FFM and one addition FFA are required in our multi-mode PGZ architecture. That is, our multi-mode PGZ architecture can solve for $t=0,1,2,3$ errors in one unified VLSI architecture, but with very small hardware overhead.

Architecture type	Number of FFM	Number of FFA
Direct implementation PGZ algorithm for $t = 3$	40	16
The derived reduced complexity PGZ for $t = 3$	21	11
The proposed Multi-mode PGZ for $t = 0, 1, 2, 3$	24	12

Table 1. Hardware complexity to perform PGZ algorithm without reusing any FFA or FFM

CONCLUSIONS

In the paper, we proposed the algorithm derivation and VLSI architecture of a multi-mode PGZ-based RS decoder. It can compute the correct error locations and error values for any t less than four, accompanied by very small complexity. Due to the help of configurable architecture, we can easily perform the RS code for different values of t without re-designing the hardware architecture.

References

[1] S. B. Wicker, *Error Control Systems for Digital Communication and Storage.* Prentice Hall, 1995.

[2] Wicker and Bhargava, Reed-*Solomon codes and applications,* IEEE Press, 1994.

[3] S. Whitaker, J. Canaris, and K. Cameron, "Reed-Solomon VLSI codec for advanced television," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, pp. 230-236, June 1991.

[4] Meera Srinivasan and Dilip V. Sarwate, *Malfunction in the Peterson-Gorenstein-Zierler Decoder*, IEEE Trans. on Information Theory, vol. 40, no. 5, September 1994.

[5] Son Le-Ngoc, Z. Young, *An approach to double error correcting Reed-Solomon decoding without Chien search*, Proceedings of the 36th Midwest Symposium, vol. 1, pp. 534-537, 1993.

[6] Kuang Yung Liu, *Architecture for VLSI Design of Reed-Solomon Decoders*, IEEE Trans. On computers, vol. C-33, no. 2, Feb 1984.

[7] L. Song and K. K. Parhi, *Low-Energy Software Reed-Solomon Codecs Using Specialized Finite Field Datapath and Division-Free Berlekamp-Massey Algorithm*, IEEE Symp. On Circuits ans Systems, June 1999.