Simplification of Procedure for Decoding Reed-Solomon Codes Using Various Algorithms: An Introductory Survey

[1]Vivek Tilavat, [2] Dr.Yagnesh Shukla
[1]PG Student, [2]Associate Professor
[1] Department of EC, SVIT, Vasad , India
[2] Department of EC, SVIT , Vasad , India
[1]vivek0880@gmail.com, [2] ybshukla2003@yahoo.com

***Abstract*— Reed-Solomon codes are very useful and effective in burst error in noisy environment. In decoding process for 1 error or 2 errors create easily with using procedure of Peterson-Gorenstein –Zierler algorithm. If decoding process for 3 or more errors, these errors can be solved with key equation of a new algorithm named Berlekamp-massey algorithm. In this paper, wide discussion of procedures of Peterson-Gorenstein –Zierler algorithm and Berlekamp-Massey algorithm and show the advantages of modified version of Berlekam-Massey algorithm with its steps.**

***Keywords*— Reed-Solomon codes, Peterson-Gorenstein-Zierler algorithm, and Berlekamp-Massy algorithm.**

I. INTRODUCTION

Each digital communication system use to transmit information source to destination in form of codeword via communication channel. These systems must be reliable and cost effective therefore some error correcting code needed. Reed-Solomon codes are most effective error correcting codes which used in communication system. Because of Reed-Solomon codes are FEC (Foreword error correcting code) that means they are not needed retransmission when errors come [1].

Reed-Solomon codes are used in many applications like in space communication links, storage devices, wireless communication, digital T.V., Broad band modems and satellite communication [2].

Reed-Solomon code has all good properties like linear, cyclic, systematic and non binary so it is used in burst errors. Means that continues 25 bits errors in code at receiver than only four symbol of byte changed and error corrected, in this code the information is taken in symbol which is made of group of bits [3].

II. REED-SOLOMON CODE

A Reed-Solomon code is a block code and specified as RS(n,k)as shown in Fig.1. The variable n is the size of the codeword with the unit symbol it is maximum with approximately 255.k is the number of the data symbols and 2t is the number of parity symbols each symbol contain s number of bits [4].

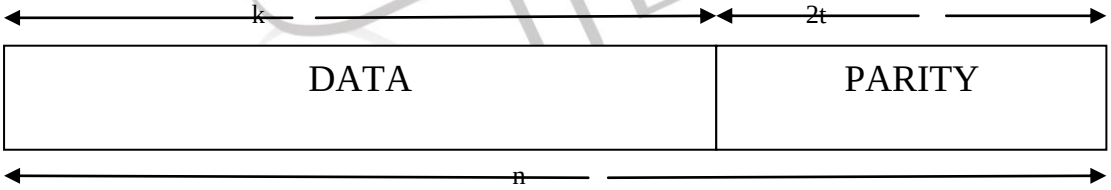

Fig.1. the structure of RS codeword

The relationship between the symbol size s and the size of codeword n is given by (1). This means that if there are s bits in one symbol, there could be $2^s - 1$ distinct symbols in one codeword, excluding the one with all zero.

$$n = 2^{s-1} \quad (1)$$

This code allows correcting up to t number of symbol errors where t is given by,

$$t = \frac{n-k}{2} \quad (2)$$

The Reed-Solomon code is defined in the Galois field [5], which contains a finite set of numbers where arithmetic operations on elements of that set will result in an element belonging to same set. Every element except zero expressed in form of primitive element α. And the non zero field elements form cyclic group defined based on a binary primitive polynomial.

An addition of two elements in the Galois field is simply the XOR operation. In case of multiplication and division of Galois field is more complex than previous. For example construction of Galois field GF(8) for primitive polynomial P(x)=$x^3 + x + 1$

base on the primitive element.

TABLE 1
Galois field GF(8)

Exponent	Polynomial	Binary
α^0	1	001
α^1	x	010
α^2	x^2	001
α^3	x+1	011
α^4	x^2 +x	110
α^5	x^2 +x +1	111
α^6	x^2 +1	101
$\alpha^7 = \alpha^0$	1	001
$\alpha^1 = \alpha^1$	x	010

The GF(8) will be the basis for the Reed-Solomon code RS(7,3).Here each symbol has 3 bits, the variables for RS code are one n=$2^3 - 1$=7, k=3 so here the input data in any 3 symbols of 3 bits like 011 001 010. And the polynomial of m(x) = $\alpha^3 x^2 + \alpha^0 x + \alpha^1$[4].

III. ENCODER

The transmitted codeword is systematically encoded and defined (3) as a function of the transmitted message m(x) with the number of parity polynomial of 2t symbols is following way [4].

$$c(x) = m(x).x^{2t} + rem\ \frac{m(x).x^{2t}}{g(x)} \quad (3)$$

Where g(x) is the generator polynomial of degree 2t and given by

$$g(x) = \prod_{i=0}^{2t-1}(x + \alpha^{mo+i}) \quad (4)$$

IV. DECODER

The Reed-Solomon decoder tries to correct errors and/or erasures by calculating the syndromes for each codeword. Based upon the syndromes the decoder is able to determine the number of errors in the received block [3].If there are errors present, the decoder tries to find the locations of the errors using the various algorithm by creating an error locator polynomial. The roots of this polynomial are found using the Chien search algorithm. Using Forney's algorithm, the symbol error values are found and corrected.

General Process of decoding

1. Syndrome Calculation
2. Determine error-location polynomial
3. Solving the error locator polynomial - Chien search
4. Calculating the error Magnitude
5. Error Correction

Here algorithms are used to find error locations and error values for various applications. When error correcting capability of Reed Solomon code is limited to one that means n-k=2 and also two error means n-k=4 at that time error evaluator polynomial is not necessary and it can be easily solved by Peterson-Gorenstein-Zierler decoding algorithm [8].

A. Peterson-Gorenstein-Zierler decoding algorithm

It is very useful for decoding 1 error or 2 errors. But errors are more than 2 then this algorithm is not useful at that time next algorithm used, here we show the procedure of decoding errors using peter-Gorenstein-Zierler algorithm [8][12].

First step of decoding is syndrome calculation for t errors are present and correct them so 2t syndromes are needed. They calculated following way

$$S(i)=r(\alpha^{mo+i}) \quad (5)$$

Here m_0=log of the initial root

Received codeword polynomial is the sum of the message polynomial and error polynomial.

$$S(i)=c(\alpha^{mo+i})+e(\alpha^{mo+i})=e(\alpha^{mo+i})=\sum_{t=0}^{t} y_t * x_t^{i+m0} \quad \text{where } (i=0\ldots\ldots\ldots 2t-1) \tag{6}$$

y_t is the error value and x_t is error location in equation. Now new term chein search ,which is important step in decoding process for finding out the location of the errors in the received codeword. This error locator polynomial $\sigma(x)$ ready then their roots gives the location of errors. Error locator polynomial given by

$$\sigma(x)=\prod_{t=1}^{t}(x+X_t)=x^t+\sigma_t x^{t-1}+\ \ldots\ldots\ldots\ldots+\sigma_{t-1}x+\sigma_t \tag{7}$$

This value is related with syndrome value values s(j) given by

$$s(t+j)+\sigma_1 s(t+j-1)+\ \ldots\ldots+s(j)\,\sigma_t=0 \tag{8}$$

It also satisfy the operations

$$\sigma_1 x^{-1}+\sigma_2 x^{-2}+\ldots\ldots+\sigma_t x^{-t}=1 \tag{9}$$

The chien sum which is sum of registers is zero then error has been located. This process for decoding one error following way,

Syndromes are given by:

$$S(0)=r(\alpha^{m_0}) \text{ and } S(1)=r(\alpha^{m_0+1}) \tag{10}$$

Now error Locator polynomial is given by:

$$\sigma(x)=x+\sigma_1$$

Some algebraic manipulation to arrive which called Newton's identities [10]

$$S(t+j)+\sigma_1\, S(t+j-1)+\cdots S(j)\sigma_t=1 \tag{11}$$

Now $S(1)+\sigma_1 S(0)=0$ where $\sigma_1=-\frac{S(1)}{s(0)}$ (12)

So calculated the error position of error using chein search means chein sum is value with zero at that place error comes. And also S(0)=0 then no error.

After that calculate the error value with

$$Y_1=S(0)\,.\,x_1^{-m0} \tag{13}$$

And this process for correcting two errors

Syndromes are given by:

$$S(0)=r(\alpha^{m_0}),S(1)=r(\alpha^{m_0+1}),\ S(2)=r(\alpha^{m_0+2}) \text{ and } S(3)=r(\alpha^{m_0+3}) \tag{14}$$

Now error Locator polynomial is given by:

$$\sigma(x)=x^2+\sigma_1.x+\sigma_2 \tag{15}$$

Some algebraic manipulation to arrive which called Newton's identities [10]

$$S(t+j)+\sigma_1\, S(t+j-1)+\cdots S(j)\sigma_t=0 \tag{16}$$

$$\begin{bmatrix} S(0) & S(1) \\ S(1) & S(2) \end{bmatrix}\cdot\begin{bmatrix}\sigma_2 \\ \sigma_1\end{bmatrix}=\begin{bmatrix}-S(2) \\ -S(3)\end{bmatrix} \tag{17}$$

So calculate σ_1 and σ_2 , after this error position of error using chein search , means chein sum is value with zero at that place error comes. And also S(0).S(2)+S(1).S(1)=0 then 1 error decoding steps follow.

After that calculate the error value Y_1 and Y_2 with equation

$$\begin{bmatrix} x_1^{m0} & x_2^{m0} \\ x_1^{m0+1} & x_2^{m0+1}\end{bmatrix}\cdot\begin{bmatrix}Y_1 \\ Y_2\end{bmatrix}=\begin{bmatrix}S(0) \\ S(1)\end{bmatrix} \tag{18}$$

B. Berlekamp Massey algorithm

Decoding for 3 or more errors present than previous algorithm not working because of matrix size becomes more, so not appropriate for these decoding. In this case upper algorithm and methods are not appropriate for pipelined design. For 3 or more error calculated that method then the matrix solution becomes more difficult and inefficient. A more efficient method is created using the Original Berlekamp Massey algorithm. In this error locator and error evaluator polynomial can be simultaneously computed. Then chein search is create and error calculated using Forney's algorithm [6][7].For this algorithm initializations are made following way[9][12]

$$\Lambda(x)^{(0)}=1,\ B(x)^{(0)}=1,\ \Omega(x)^{(0)}=0,\ A(x)^{(0)}=x^{-1},\ L^{(0)}=0 \tag{19}$$

$\Lambda(x)$ is error locator polynomial, $B(x)$ is the error locator support polynomial, $\Omega(x)$ is error evaluator polynomial, and $A(x)$is the error evaluator support polynomial. L is integer variable.

The syndrome represented by the syndrome polynomial:

$$S(x)=\sum_{j=0}^{2t-1} s_j x^j \tag{20}$$

The algorithm iterates for 2t steps. At the (k+1) step, calculate the following term discrepancy:

$$\Delta^{(k+1)}=\sum_{j=0}^{L(K)}\Lambda_j^{(k)} s_{k-j} \tag{21}$$

Then

$$\Lambda(x)^{(k+1)} = \Lambda(x)^{(k)} - \Delta^{(k+1)} B(x)^{(k)} x \quad (22)$$

$$\Omega(x)^{(k+1)} = \Omega(x)^{(k)} - \Delta^{(k+1)} A(x)^{(k)} x \quad (23)$$

If $\Delta^{(k+1)} = 0$ or 2L>k then

$$B(x)^{(k+1)} = B(x)^{(k)} x \quad (24)$$

$$A(x)^{(k+1)} = A(x)^{(k)} x \quad (25)$$

$$L^{(k+1)} = L^{(k)} \quad (26)$$

Otherwise

$$B(x)^{(k+1)} = \frac{\Lambda(x)^{(k)}}{\Delta^{(k+1)}} \quad (27)$$

$$A(x)^{(k+1)} = \frac{\Omega(x)^{(k)}}{\Delta^{(k+1)}} \quad (28)$$

$$L^{(k+1)} = k + 1 - L^{(k)} \quad (29)$$

One drawback is that to perform galois field inversion in every iteration is very difficult.

An improvement is the inversionless Berlekamp Massey algorithm [10][11].In this upper drawback is eliminated but its output give only one result of error locator polynomial.

Here in modification of this algorithm initialization is

$$l^{(0)} = 0 \text{ , and } \gamma^{(k)} = 1 \text{ if } k \leq 0 \quad (30)$$

$$\mu(x)^{(0)} = 1, \lambda(x)^{(0)} = 1 \quad (31)$$

Here calculate following term

$$\boldsymbol{\delta}^{(k+1)} = \sum_{j=0}^{L(K)} \mu_j^{(k)} s_{k-j} \quad (32)$$

Then

Error locator polynomial $\mu(x)^{(k+1)} = \gamma^{(k)} \mu(x)^{(k)} - \boldsymbol{\delta}^{(k+1)} \lambda(x)^{(k)} x$ (33)

If $\boldsymbol{\delta}^{(k+1)} = 0$ or $2l^{(k)} > k$ then

$$\lambda(x)^{(k+1)} = \lambda(x)^{(k)} x \quad (34)$$

$$l^{(k+1)} = l^{(k)} \quad (35)$$

$$\gamma^{(k+1)} = \gamma^{(k)} \quad (36)$$

Otherwise

$$\lambda(x)^{(k+1)} = \mu(x)^{(k)} \quad (37)$$

$$l^{(k+1)} = k + 1 - l^{(k)} \quad (38)$$

$$\gamma^{(k+1)} = \boldsymbol{\delta}^{(k+1)} \quad (39)$$

The error evaluator polynomial is calculated by following equation [6].

$$\mu(x).S(x) = \Omega(x) \bmod (x^{2t}) \quad (40)$$

In this algorithm need extra clock cycles needed to perform such a polynomial multiplication so longer latency required which is also drawback.

This algorithm can be extended to yield both this error locator and error evaluator polynomial computed at same time, and without using galois field inversion. And also upper polynomial multiplication is reducing with it [8].

In this algorithm of following steps are added in inversion less berlekamp massey algorithm

$$\Omega(x)^{k+1} = \gamma^k . \Omega(x)^k - \delta^{k+1} . a(x)^k . x \quad (41)$$

The initial conditions are same as that original first Berlekamp Massey algorithm those are

$$\Omega(x)^0 = 0, \; a(x) = x^{-1} \quad (42)$$

$\Omega(x)$ is the error evaluter polynomial and $a(x)$ is the error evaluator support polynomial. And the point both error locator and error evaluator polynomials have been computed. The Chein search will find the zeros of the error locator polynomial. The next step is to correct the errors using Forney algorithm [12],

$$e_{t=} \alpha^t \frac{\Omega(\alpha^t)}{\Lambda'(\alpha^t)} \quad (43)$$

V. CONCLUSION

From introductory survey of various algorithm for decoding Reed Solomon codes, we saw that small errors like 1error or 2 error present at receiver side then peter-Gorenstein-Zierler algorithm is used, it solved with easily matrix equations. But the errors are more at the receiver side then this algorithms key equation highly structured matrix. So another Berlekamp Massey algorithm used. And its modification used for remove the galois field inversion which is most useful for creating VHDL codes and also useful for implemented on FPGA.

VI. REFERENCES

[1] Priyanka Shrivastava and Uday Pratap Singh, " Error correction using Reed Solomon Codes." ijarcsse, Vol. 3, Issue 8, August 2013

[2] Tsung-Chang Lin ,Trieu-Kiren Truong Hsin-Chin chang and Hung-Peng Lee, "A Future Simplification of Procedure for DecodingNonsystematic Reed Solomon codes Using the Berlekamp-Massey Algorithm.", IEEE Transactions on communications , Vol 59,No.6 ,June 2011.

[3] Bernard skalar, "Digital Communications: Fundamentals and application",2nd addition .

[4] Kenny Chang wai and Dr. Shanchieh Jay Yang. "Field Programmable Gate Array Implementation", IJACSA, Vol.5, No. 3, October 2011.

[5] G.Alquist, B.Nelson and M.Rice, " Synthesis of small and fast finite field multipliers for field programmable gate arrays, In processings of 5th Annual Military and Aerospace Programmable Logic Device International Conference ,September 2002.

[6] I.S. Reed, M.T. Shih, T.K. Truong , "VLSI design of inverse-free Berlekamp-Massey algorithm", IEE PROCEEDINGS-E, Vol. 138, No. 5, September 1991.

[7] V.Glavac and M.R.Soleymani, "An Extended Inversion-less Massey-Berlekamp Algorithm", 2000 Conference on Systemics,Cybernetics and Informatics, SCI, Orlando,Florida,July 23-26 ,2000.

[8] D.Gorenstein And N.Zierler, "A Class of Error Correcting Codes in p^m symbols", Journal of the Society of Industrial and Applied Mathemetics, Vol9 , June 1961.

[9] Jurgen Freudenberger and Jens Spinner HTWG Konstanz, "Mixed serial/parallel hardware implementation of the Berlekamp-Massey algorithm for BCH decoding in Flash controller applications", signals, systems and electronics (ISSSE) IEEE conference publication, 2012.

[10] Arnold M.Michelson and Allen H.Levesque, "Error Control Techniques for Digital Communication", John Wiley & Sons, 1985.

[11] Hazem A. Ahmed, Hamed Salah, Tallal Elshabrawy, Hossam A. H. Fahmy , "Low Energy High Speed Reed-Solomon Decoder Using Two parallel Modified Evaluator Inversion less Berlekamp-Massey", signals, systems and computers(ASILOMAR) IEEE, 2010.

[12] Richard E. Blahut, "Algebraic Codes for Data Transmission Cambridge University press", 2003.