ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

A Survey on Reed Solomon Encoder and Decoder

Shital M. Mahajan[1], Piyush M. Dhande[2]

M. Tech Student, Dept. of E&TC, D.M.I.E.T.R. Sawangi, Wardha, India[1]

Asst. Professor, Dept. of E&TC, D.M.I.E.T.R. Sawangi, Wardha, India[2]

ABSTRACT*:* In digital communication system forward error correction codes are generally used for error detection and correction. The reed Solomon (RS) is most commonly used FEC code as it detects and corrects the errors on symbol level. Reed Solomon codes are linear block code. It works on information by dividing the message stream into blocks of data and adds the redundancy depending on the current input. Galois field is used for encoding and decoding the RS code. The Reed Solomon encoder uses Galois field multiplier and linear feedback shift register to encode the data. While decoder having syndrome calculator and key equation solver blocks, which are used to determine the magnitude and location of the errors in received codeword. For this purpose decoder uses different decoding techniques like Berlekamp Massey, Euclid, Chien Search, Forney algorithm. This paper proposed the review of Reed Solomon encoder and decoder.

KEYWORDS: Forward error correction (FEC) code, Linear Block Code (LBC), Redundancy, Galois Field, Berlekamp-Massey, Euclid, Chien Search, Forney.

I. INTRODUCTION

In digital communication system, information or data gets corrupted during transmission. So that transmitter has to send data again and again, this causes loss of resources and thus results in burden on transmitter. So that it becomes necessary to detect and correct errors during communication. Error correcting codes are able to detect and correct errors in communication system. Reed Solomon codes are the most commonly used error correcting codes. Reed Solomon codes were discovered by Irving Reed and Gustave Solomon, in Lincoln Laboratory of Massachusetts Institute of Technology in 1960. Firstly the RS codes were introduced in a paper named Polynomial codes over certain finite fields in 1960 for burst error correction [2].

Reed-Solomon code is a well known non-binary linear block code. Reed Solomon codes operate on the information by dividing the original message into blocks of data and adds redundancy on each block depending only on the current inputs. Figure 1 shows the block diagram of communication system [1].

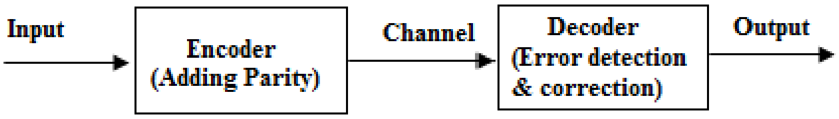

Fig.1. Block diagram of communication system

II. REPRESENTATION OF REED SOLOMON CODES

Reed Solomon codes are the forward error correcting code that can be given by RS (n, k), where, n is the block length and k is the message length. The difference in block length and message length called as parity bits. And it is denoted by 2t. Reed Solomon codes are able to correct up to t errors. It corrects the error on symbol level. Length of symbol is denoted by m. The fig.2 shows the codeword of Reed Solomon code [4].

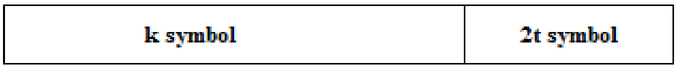

Fig. 2 Codeword generated by reed Solomon encoder

Input message symbols to RS encoder	: k
Codeword generated by RS encoder	: n
Parity symbols	: 2t
Size of each symbol	: m
Error correcting capability	: t

III. LITERATURE SURVEY

The most important part of reed Solomon encoder is the linear feedback shift register that is implemented using VHDL. The decoder contains syndrome calculator, Berlekamp Massey Algorithm and Chien search algorithm. Pipelining is introduced in Chien-Search component of decoder to improve the frequency of Reed-Solomon codes. Thus the frequency of encoder increases from 308.5 MHz to 439.165 MHZ and that of the decoder from 145.6 MHz to 152.592MHz. The Reed-Solomon code RS (255,239) for wireless network 802.16 is being synthesized using VHDL on Xilinx 12.4 ISE and simulated on ISE simulator [1].

Modified-Berlekamp-Massey (mBM) algorithm for non binary Galois field which eliminates the inverses. The operation needed for Berlekamp-Massey algorithm and modified Berlekamp-Massey algorithm are same. They need extra multiplications in the modified method and also the division operation needed in the original method. The division operation used in the original BM method required a table-lookup to find an inverse element, which can be time consuming process. While in modified Berlekamp-Massey algorithm, one extra preloaded register is used which stores syndromes codeword follows the modified structure. So that, VLSI structure of modified Berlekamp-Massey algorithm is simple and suitable for decoding of Reed-Solomon codes [5].

High speed architecture gives the inversion less Berlekamp-Massey algorithm version of Berlekamp-Massey architectures. This new algorithm is used to compute the error locator and the error evaluator polynomials. It reduces the critical path delay with respect to the previously available Berlekamp-Massey algorithm as it uses the less number of multipliers and adders. This architectures have comparable data rates and less gate complexity [6].

Self-checking RS encoder and decoder architectures removes the errors or faults present in RS CODEC which compromise the reliability of whole system. The Reed Solomon encoder architecture uses some properties of the arithmetic operations in Galois field GF (2^m). These properties are related to binary representation of the elements of the Galois Field. At the Reed Solomon decoder, the implicit redundancy of this the received codeword under suitable assumptions explained. It allows implementing concurrent error detection schemes which is useful for a wide range of different decoding algorithms with no effect on the decoder architecture [8].

IV. REED SOLOMON ENCODER

A Reed Solomon codeword can be calculated from input message symbols by obtaining a generator polynomial. A generator polynomial depends on the order of the Galois field over which RS code has been defined and numbers of error to be detected and corrected. The primitive polynomial is given by the order of the Galois field. One of the primitive polynomial in GF (2^m) field is $x^8+x^4+x^3+x^2+1$. Let α be the primitive element in the Galois field in GF (2^m). Then the generator polynomial for correcting t errors is as given by [9]

$$g(x)= \prod_{i=1}^{2t}(x+\alpha i) \quad \ldots\ldots\ldots\ldots(1)$$

In concurrent error detection in Reed–Solomon encoders and decoders, increases the codeword length keeping message length constant by using GF (2^m) with m>8. This increases the complexity in the encoding and decoding. To overcome this problem author proposed parity sharing (PS) Reed Solomon code which is obtained by concatenating two different RS code. This also improves the speed and throughput [8]. In architecture for decoding adaptive Reed Solomon codes

with variable block length, proposed that the message length changes while code length remains constant. Author used adaptive FEC technique and different code rate which affects on error correcting capability [14].
Figure 3 shows the block diagram of reed Solomon encoder [4].

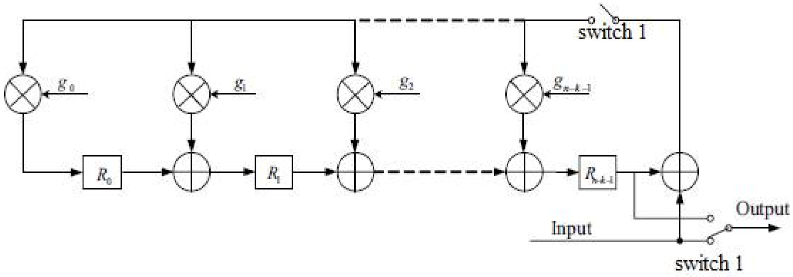

Fig.3. Block diagram of RS Encoder

Code specification:

Block length:	n = 255
Message Length:	k = 239
Symbol width:	m=8
Field generator polynomial:	$P(x) = X^8+X^4+X^3+X^2+1$
Code generator polynomial:	$G(x) = (X+\alpha)(X+\alpha^2)(X+\alpha^3)\ldots\ldots\ldots\ldots(X+\alpha^{2t})$

V. REED SOLOMON DECODER

The encoded data given to the decoder. At the decoder syndrome gets calculated also error magnitude and error locations are found out. Figure 4 shows the block diagram of Reed Solomon decoder [3].

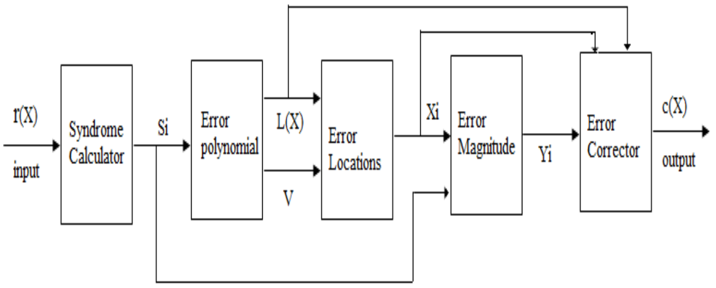

Fig 4.Block diagram of RS Decoder

Where,

r(x) -----	Received codeword
Si -----	Syndromes
L(x) -----	Error locator-polynomial
Xi -----	Error locations
Yi -----	Error magnitudes
c(x) -----	Recovered code word
v -----	Number of errors

At the decoder received codeword is the original transmitted codeword with error.

$$r(X) = c(X) + e(X) \qquad \ldots\ldots.(2)$$

To obtained original codeword we can use the following steps:

A. *Syndrome Calculation*

This calculation is equivalent to parity calculation. As Reed-Solomon codeword has 2t syndrome values that depend only on errors not on the transmitted code word. The syndromes can be calculated by dividing received codeword by generator polynomial. Remainder obtained in this calculation is our syndrome. If remainder is zero then no error present in the received codeword and if remainder is obtained then error present in that codeword. Fig, 5 shows the syndrome generator architecture [9].

$$\frac{r(X)}{g(X)} = \mathrm{P(X)} + \frac{s(X)}{g(X)} \quad \ldots\ldots\ldots.(3)$$

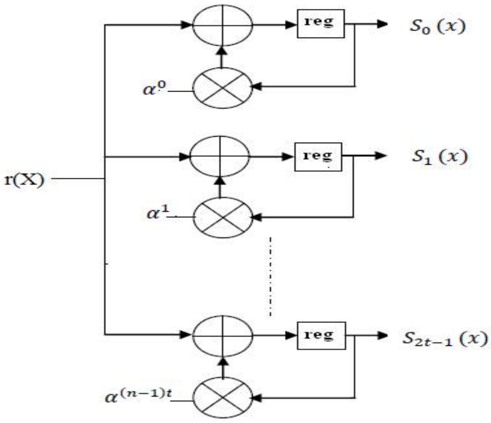

Fig.5. Syndrome generator architecture

Syndrome polynomial is given by

$$\mathrm{S(X)} = S_0+S_1X^1+S_2X^2+\ldots\ldots..S_{(2t-1)}X^{(2t-1)} \quad \ldots\ldots\ldots(4)$$

B. *Determination of error-locator polynomial*

After calculation of the syndrome polynomial, next step is to compute the error values and their locations in. This stage contains the solving of the 2t values of syndrome polynomials from the previous stage. These polynomials have total **v** unknown terms, where v is the number of unknown errors prior to decoding.

C. *Solving the error locator polynomial*

After getting the error locator and error evaluator polynomial the next work to be done is to evaluate the error polynomial and to obtain its roots. The roots are obtained will now point to error location in the received code. RS decoding generally use the Chien search algorithm to implement the same.

D. *Calculate error value*

Once the error location find out, the next step is to use the syndrome values and the error polynomial roots to derive the error values. Use Forney's Algorithm for this purpose. It is an effective way of performing a matrix inversion, and it involves two main stages.

E. *Error correction*

If the error symbol has any number of set bit, it means that the corresponding bit in the received code is having an error, and must be inverted. To convert this correction process each of the received symbol is read again, and at the each error location the received symbols are XOR with the error symbols. Thus the decoder corrects any errors as the received codeword is being read out from it.

F. Key Equation Solver architecture of decoder

Table-1 summarizes the complexity of the various published Key Equation Solver(KES) architectures [6]. It can be seen that, in comparison to the conventional iBM architecture (Berlekamp's version), the riBM systolic architecture requires more multipliers and more multiplexers. All two architectures require the equal numbers of latches and adders and all two architectures require *2t* cycles to solve the key equation for t error correcting code. The riBM architecture requires considerably more gates than the conventional iBM architecture called as Blahut's version, but also require only *2f* clock cycles as compared to the *31* clock cycles required by the latter. Furthermore, since the critical path delay in the riBM architecture is less than half of the critical path delay in either of the iBM architectures, [6] concludes that the riBM architecture significantly reduces the total time required to solve the key equation and thus obtained higher throughput with only a modest increase in gate count. More important, the regularity of the riBM architecture creates the potential for automatically generating regular layouts via a core generate or with predictable delays for various values of *f* and *m* [6].

Table 1: Comparative study of Algorithms used by RS Encoder and Decoder

Architecture	Adders	Multipliers	Latches	Muxes	Clock	Critical path delay
Euclidean[11]	4t+2	8t+8	4t+4	8t+8	2t	Tmul+Tadd+Tmux
IBMA [12]	2t+1	3t+3	4t+2	t+1	2t	2Tmul+[log$_2$(2t+2)]Tadd
RiBMA [5]	3t+1	6t+2	6t+2	6t+1	2t	Tmul+Tadd
eIBMA [6]	2t	3t+2	2t+2	t+1	2t	2Tmul+[log$_2$(2t+2)]Tadd
ePIBMA [10]	2t+1	4t+2	4t+2	6t+3	2t	Tmul+Tadd
ePIBMA (2-folded)	t+1	2t+2	6t+6	5t+5	4t	Tmul+Tadd+Tmux

VI. PROBLEM FORMULATION AND PROPOSED WORK

In the existence system reed Solomon encoder and decoder has been developed for high speed low error communication. but the computational complexity of reed Solomon encoder and decoder is high. And thus limit the speed of encoding and decoding there by introducing inherent delays in the system. this also causes errors in the system for fast changing input and outputs.

To overcome this problem we proposed a pipelined reed Solomon encoder and decoder. In this arrangement the input is firstly stored in the pipelined buffer which will stored the input until the reed Solomon encoder is ready for the operation. Which reduces the loss of input for fast changing signals. The output of the encoder is again same in pipelined buffer so that the output device can copy that data when it is ready. Similarly decoder also has input and output pipelined buffer in order to improve the speed and reduce error in the system.

This sort of pipelined arrangement will improve the throughput of the system with a minimum overhead of area and power.

VII. CONCLUSION

This work introduced the review of reed Solomon encoder and decoder. In this paper we have mentioned various technique used to calculate errors and also compared various algorithms used by Reed Solomon encoder and decoder.

REFERENCES

1.Priyanka Dayal, Rajeev Kumar Patial, " Implementation of Reed- Solomon CODEC for IEEE 802.16 network using VHDL code", 2014 International Conference on Reliability, Optimization and Information Technology ICROIT 2014, India, pp 452-455, Feb 6-8, 2014

2.S.Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields", SIAM Journal of Applied Mathematics, vol. 8, pp. 300–304, june 1960.

3.Priyanka Shrivastava, Uday Pratap Singh, "Error Detection and Correction Using Reed Solomon Codes", International Journal of Advanced Research in Computer Science and Software EngineeringVolume 3, Issue 8,pp 965-969, August 2013.

4.P. Ravi Tej, Smt. K. J hansi Rani, "VHDL Implementation of Reed Solomon Improved Encoding Algorithm", International Journal of Research in Computer and Communication Technology, Vol 2,pp 435-439, August -2013

5.I S. Reed, M. T. Shih, T.K. Truong, "VLSI design of inverse-free Berlekamp-Massey algorithm," Proceeding Inst. Elect. Eng., pt. E, vol. 138, pp. 2015