International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 2, February - 2014

Design of Reed Solomon Encoder and LCC Decoder Based on Unified Syndrome Computation

Jeseena S
PG Student/Department of ECE
TKM Institute of Technology, Kollam,
Kerala, India

Suby Varghese
Assistant Professor/Department of ECE
TKM Institute of Technology, Kollam
Kerala, India

***Abstract*— In high speed communication system, Reed-Solomon (RS) codes are widely used to provide error protection especially against the burst errors. Reed Solomon codes are non-binary, linear block based forward error correction code. In this paper a Reed-Solomon encoder and an efficient decoder having better error correcting capability has been designed. The Reed Solomon encoding is done by adding redundant information to the data so that the receiver can correct errors caused by channel noise. The error correcting capability can be improved by incorporating the reliability information from the channel into algebraic soft decision Low Complexity Chase (LCC) decoding algorithm. The RS decoder makes use of the LCC decoding based on hard decision decoding (HDD) with the Reduced inversion-less Berlekamp-Massey (RiBM) algorithm.The coding is done in VHDL language, synthesized using Xilinx ISE 13.2 and simulated using ModelSim.**

Keywords* — *Hard decision decoding (HDD), Algebraic Soft-Decision Decoding (ASD), Low Complexity Chase (LCC), Unified syndrome computation

IJERT

I. INTRODUCTION

Digital communication system is used to transmit an information bearing signal from the source to a destination through a communication channel. Channel coding is an important signal processing operation for the efficient transmission of digital information over the channel. The power of Forward Error Correction (FEC) is that the system can, without retransmissions, find and correct limited errors caused by a transmission or storage system. Reed-Solomon (RS) code is a well-known non-binary linear block FEC code. It is popularly used for error correction in many applications like storage devices (CD, DVD), wireless communications, high speed modems and satellite communications.

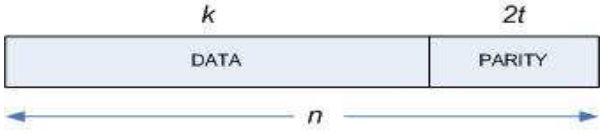

Fig. 1 Reed Solomon Codeword

RS code can be specified as RS (n, k) with m-bit symbols, where $n=2^m-1$ as shown in Figure 1.The variable n is the size of the codeword with the unit of symbols, k is the number of data symbols and 2t is the number of parity symbols. A predetermined sized block of data (k bytes) is encoded so that the result is a data block of size n, where n>k. This n block contains the k original data bytes, along with n-k parity bytes, representing the redundancy in the signal, for transmission over the noisy channel. Within the block, the RS algorithm works on multiple bits of data at a time, typically a byte. Each byte is a symbol, and the nature of the RS algorithm allows for the correction of whole symbols, as opposed to correcting individual bits. This means that the RS decoder can correct up to 't' symbols that contain errors in a code word, where 2t=n-k. This is the particular characteristic which allows RS codes to be effective at correcting burst errors in addition to random errors.

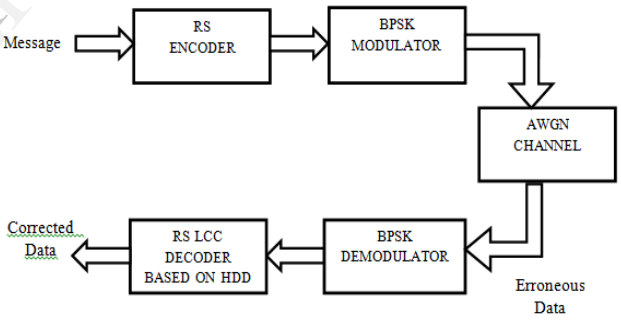

Fig. 2 System model

The proposed system consists of four main blocks: RS encoder, BPSK modulator, AWGN channel, BPSK demodulator and RS decoder as shown in figure 2. The RS encoder unit takes the incoming data and adds redundancy to obtain the codeword. The encoded message codeword are fed into a modulator, which maps each of the information sequence into signal waveforms. The channel over which the waveforms are transmitted will corrupt the waveforms by adding symmetric additive white Gaussian noise (AWGN). At the receiver, the digital demodulator processes the channel corrupted signal waveforms and reduces it into a sequence of data symbols. Among the two Reed-Solomon decoding algorithms ASD has significant coding gain over HDD. Comparing the different ASD algorithms the Low Complexity Chase (LCC) decoding is found to have less computation complexity with similar or higher coding gain. The LCC algorithm consists of three main steps: multiplicity assignment,

interpolation and factorization, where interpolation stage involves major part of the computation. So instead of employing the sophisticated interpolation algorithm, the LCC decoding is done based on the HDD, to get the error locator and error evaluator polynomial.

This paper is organized as follows. Section 2 gives a brief overview of RS encoder and its corresponding architecture. In Section 3, Reed Solomon LCC decoder based on HDD is discussed. In section 4, the simulation results of the encoder and decoder sections were discussed. Conclusions and ongoing works are discussed in section 5.

II. REED SOLOMON ENCODER

The RS encoder takes a block of symbols and adds extra parity check symbols. A RS codeword can be computed from input message symbols by employing 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 corrected.The generating polynomial for most conventional RS code (n, k) takes the form

$$g(X) = g_0 + g_1X \ldots\ldots + g_{2t-1} X^{2t-1} + X^{2t} \quad (1)$$

Since the degree of generating polynomial is 2t, there are 2t roots of generating polynomial, which are the successive powers of α, where α is the generator of Galois field GF (2^m). Each symbol is considered as an element of Galois Field where α is the primitive element. Hence the roots of generating polynomial can be designated as α, α^2, α^3… α^{2t}. We can start the root with any desired power of α.

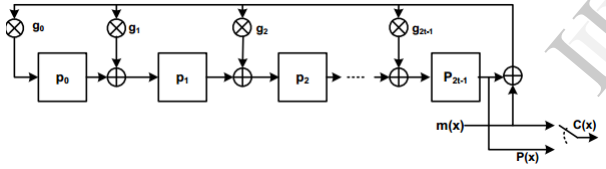

Fig. 3. Systematic encoding using LFSR

Consider RS (7,3) code as an example. The number of parity bits in the code is n-k=4. Hence we can describe the generating polynomial as given in equation 2

$$g(X)=(X-\alpha)(X-\alpha^2)(X-\alpha^3)(X-\alpha^4) \quad (2)$$

In Galois field arithmetic, addition and subtraction are considered as XOR operation, therefore g(X) can be expressed as

$$g(X) = \alpha^3 + \alpha^1 X + \alpha^0 X^2 + \alpha^3 X^3 + X^4 \quad (3)$$

The parity polynomial, p(X) is appended to the leftmost (n-k) stages of the codeword polynomial U(X). Therefore message polynomial, m(X) is multiplied with X^{n-k} to right-shift m(X) by (n–k) positions. Then X^{n-k} m(X) can be expressed as:

$$X^{n-k} m(X) = q(X)\, g(X) + p(X) \quad (4)$$

where q(X) and p(X) are quotient and remainder polynomials, respectively. As in the binary case, the remainder is the parity. Equation 4 can also be expressed as:

$$p(X) = X^{n-k} m(X) \text{ modulo } g(X) \quad (5)$$

The resulting codeword polynomial, U(X) can be written as

$$U(X) = p(X) + X^{n-k} m(X) \quad (6)$$

III. REED SOLOMON DECODER

This method combines the hard-decision decoding techniques with the LCC decoding method to decode the RS codes using channel information. The block diagram of Reed-Solomon LCC decoder based on HDD is shown in figure 4. The steps of the decoder are: multiplicity assignment, unified syndrome computation, Key Equation Solver (KES), Chien Search for error location calculation, decoding failure detection, Forney's algorithm for error value calculation and error correction. The LCC decoding process creates 2^η different test vectors using the reliability information from the received points (channel). A unified syndrome computation algorithm is used in which the syndromes of one test vector can be obtained from syndromes of previous test vector. If an error is detected, the process of correction begins by locating the errors first. Generally Berlekamp-Massey Algorithm is used to calculate the error locator polynomial. The precise location of the errors is calculated by using Chien search algorithm. Then magnitude of the errors is calculated using Forney algorithm. In this HDD-based LCC decoding algorithm, the test vectors are selected for correction during the decoding on occurrence of the HDD failure. The magnitude of the error is added to the received codeword to obtain a correct codeword.

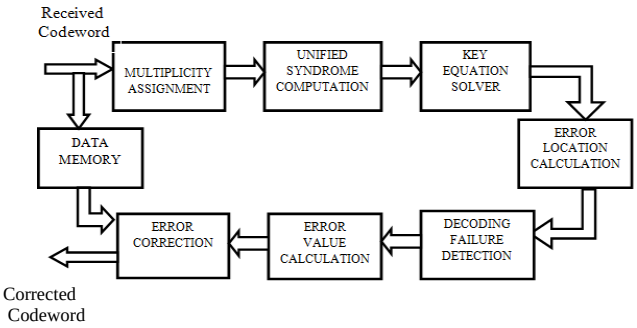

Fig. 4. RS LCC decoder based on LCC

A. *Multiplicity Assignment*

The LCC decoding process creates 2^η different test vectors using the reliability information from the received points, where $\eta<2t$. To create the test vectors a ratio between the probability of the hard-decision symbol and the second-best decision is established. This ratio indicates how good the hard decision is. The desired probability ratio for the received message polynomial r(x) for i = 0 to n-1 is

IJERT

$$\pi_{1Xn} = \left[\frac{P\left(r_0 | P\left(r_{0\_HD}\right)\right)}{P\left(r_0 | P(r_{0\_2HD})\right)} \cdots \frac{P\left(r_{n-1} | P\left(r_{(n-1)\_HD}\right)\right)}{P\left(r_{n-1} | P(r_{(n-1)\_2HD})\right)}\right] \quad (7)$$

Where $ri_{\_HD}$ is the hard decision of the symbol ri, and $ri_{\_2HD}$ its second more reliable decision. Corresponding to the η points with the worst probability ratio (between the hard decision and second-best decision), a set of 2^{η} combinations called test vectors are created by selecting the hard-decision symbol (ri) or the second-best decision ($ri_{\_2HD}$) in the η less reliable points. Once the test vectors are created, it is required to select one of them, and correct the errors of the message, if it is possible.

Now just η symbols have to be marked with zero multiplicity to create the 2^{η} test vectors, as the rest of symbols are marked with one multiplicity. To classify the multiplicity of the points, the probability ratio for the received message polynomial r(x) are ordered, and the η least values are marked as zero multiplicity: the η points with the least values will be marked as the points with different values for the test vectors.

B. Unified Syndrome Computation (USC) Architecture

The architecture of the USC consists of the syndrome calculation architecture and the syndrome update architecture. They are separated into different pipelining stages. For the syndrome calculation architecture, it is the same as ordinary syndrome architecture in HDD decoders as shown in Figure 5.

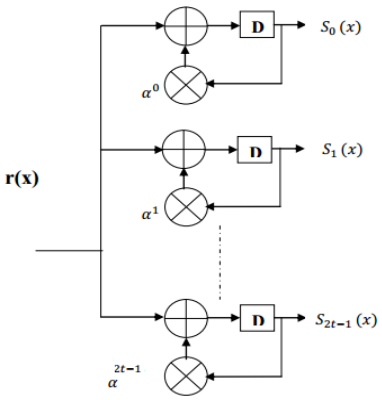

Fig. 5. Syndrome Calculation

The syndrome consists of n–k symbols and the values are computed from the received code word. The syndrome depends only on the error vector, and is independent of the transmitted code word. The method typically used to directly evaluate the received code word R(x) at each α^{n-k} is shown in figure 5. So the syndrome values can be obtained by substituting $x = \alpha^i$ in the received polynomial.

$$S_i = R(\alpha^i) \quad \text{for } i=0,1,\ldots..n-k \quad (8)$$

The syndromes of the first test vector are calculated and stored in registers from S_1 to S_{2t}, and later pass the syndrome update module in order. At the same time, the positions of η unreliable points with their r_{HD-i} and r_{2HD-i}, i ε Z , are also stored.

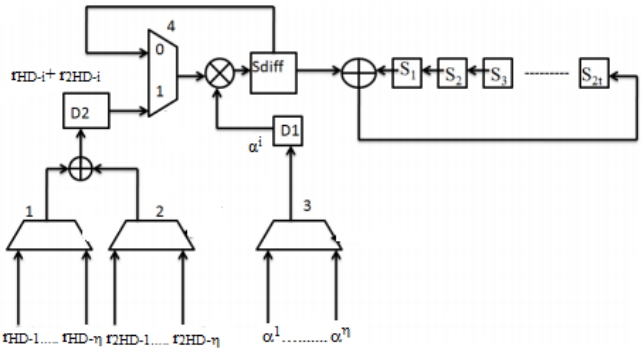

Fig. 6. Syndrome Update Architecture

In update architecture, the multiplexer 1, 2 and 3 choose $r_{HD-\varepsilon}$, $r_{2HD-\varepsilon}$ and α^{ε} of one unreliable point in Z. This selected point is the only different point between the current test vector and the next one and its position is denoted as ε. The difference of $r_{HD-\varepsilon}$ and $r_{2HD-\varepsilon}$ is calculated by sending them into an GF (2^3) adder and is multiplied by α^{ε} to get $(r_{HD-\varepsilon}+ r_{2HD-\varepsilon})\ \alpha^{\varepsilon}$. The value in S_{diff} , which is $(r_{HD-\varepsilon}+ r_{2HD-\varepsilon})\ \alpha^{\varepsilon}$ is added to the value in S1, then the sum is fed back to the register queue to update the syndromes. Also $(r_{HD-\varepsilon} + r_{2HD-\varepsilon})\ \alpha^{2\varepsilon}$ is calculated and added to S2. In this way, the stored syndromes of the first test vector can be updated by adding the corresponding syndrome difference to each item in the syndromes. Finally all the syndromes shift to the adder to finish the complete update.

C. Key Equation Solver

When all the syndrome values are zero then the received codeword is a correct codeword. If any of the syndrome value is nonzero then there is error in the received codeword. These syndrome values are then given as the input of the RiBM algorithm.

$$\lambda(x)\ S(x) = \Omega(x) \bmod x^{2t} \quad (9)$$

$$\text{where} \quad \lambda(x) = 1+\lambda_1 x + \ldots.. + \lambda_e x^e \quad (10)$$

$$\omega(x) = \omega_0 + \omega_1 x + \ldots.. + \omega_{e-1} x^{e-1} \quad (11)$$

This Key Equation Solver (KES) block gives two polynomials. The error locator polynomial computation and error evaluator polynomial computation takes place in this RiBM algorithm and performs the evaluation of the error locator polynomial coefficients λ(x) and error evaluator polynomial coefficients ω(x) thereby correct the errors as the received word is being read out of the decoder.

D. Error Location Calculation

The locations of the errors are determined based on the error locator polynomial, λ(x). Chien search is used for finding roots of the error locator polynomial. In this process α^{-j} ($0 \le j \le n-1$) is given as input to the key equation to calculate the polynomial λ(x). If $\lambda(\alpha^{-j})$ is equal to 0, then α^{-j} is an error

IJERT

location.

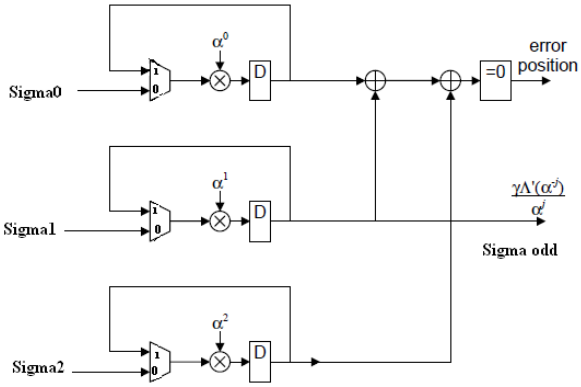

Fig. 7. Architecture for Chien-Search Algorithm

E. Decoding Failure Detector

After error location calculation, the number of zeros/roots of this step is compared with degree of λ(x). If both are equal then corresponding test vector is selected for Forney's algorithm. It means that the first vector which is free from decoding failure will be used to make a decoding of the received message. Therefore, if one of the 2^{η} test vectors has less than t +1 errors, then the decoder will be able to correct them in all cases.

F. Error Value Calculation

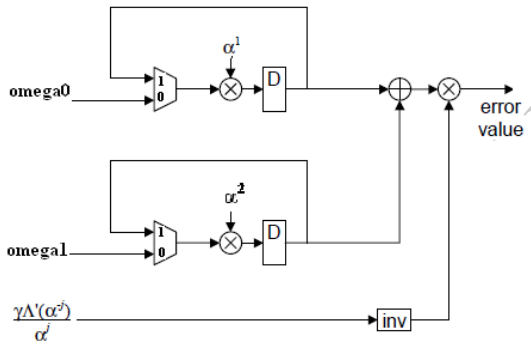

Fig. 8. Architecture for Forney algorithm

Forney Algorithm is used for evaluating the error values. The error position and the coefficients of ω(x) is taken here as input. It is also using the Galois field multiplier as the Chien search algorithm. If the error number is equal to or less than t, λ(x) and ω(x) will be obtained by the above equations 10 and 11 and the error value may be expressed by Forney's formula as,

$$Y_i = \frac{x\ \omega(x)}{\lambda_{odd}(x)}\Bigg|_{x=\alpha^{-j}} \qquad (12)$$

G. Error Correction

The output of the chien/forney block is the error vector. This vector is the same size as the codeword. The vector contains non zero values in locations that correspond to errors. Because the error vector is generated in the reverse order of the received codeword, a FIFO must be applied to either the received codeword or the error vector to match the order of the bytes in both vectors. The output of the adder is the decoder's estimate of the original codeword.

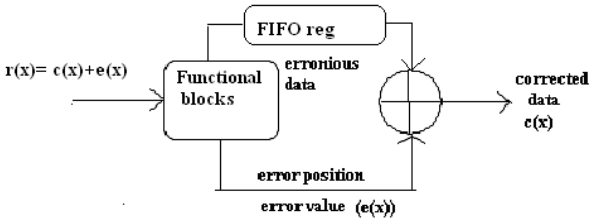

Fig. 9. Error Correction module

IV. Simulation Results

The design entry is modelled using VHDL in Xilinx ISE Design Suite 13.2 and the simulation of the design is performed using Modelsim to validate the functionality of the design.

Here a (7,3) Reed-Solomon code is designed using VHDL, for which the number of message symbols is 3 and total number of symbols in the codeword is 7. Each symbol contains 3 bits (m=3) and error correcting capability of the code is 2 symbols (t=2). So we have to add 2t i.e. 4 parity symbols. Figure 10 shows the simulation output of (7,3) RS encoder. Message is shown as 'm' and 4 check symbols (p1, p2, p3, p4) are attached with the input message 'm'.

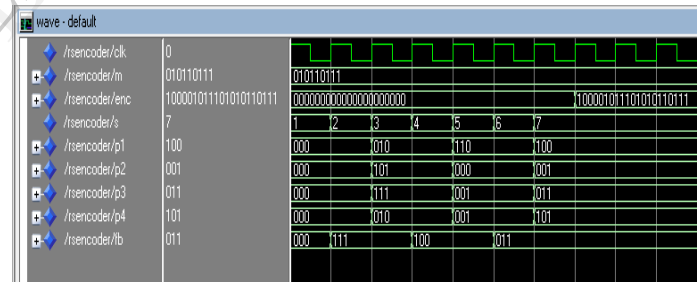

Fig. 10. RS encoder output

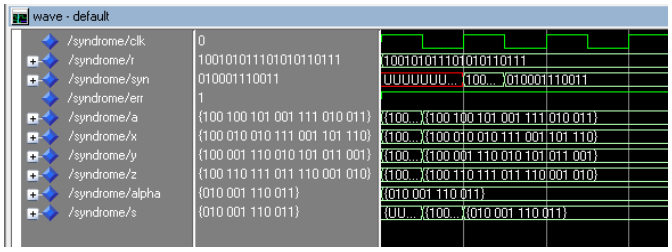

Fig. 11. Syndrome Computation output

Figure 11 shows the output of syndrome computation block. If there is no error in the received codeword 'r' then syndrome value is zero, 'err' signal will be low. If erroneous data is received, then 'err' signal is high and it indicates presence of errors and rest of the blocks will be active.

IJERT

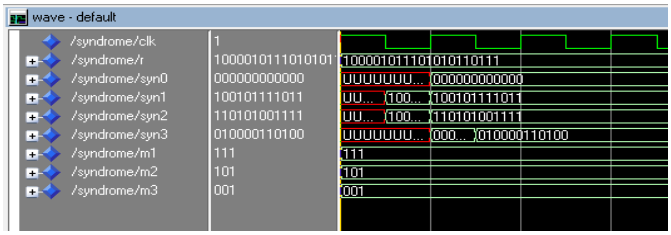

Fig. 12. Syndrome Update output

Figure 12 shows the syndrome update output of all the test vectors. By knowing the multiplicities of the received test vector and their positions, the syndrome computation of η unreliable points thereby all test vectors can be obtained.

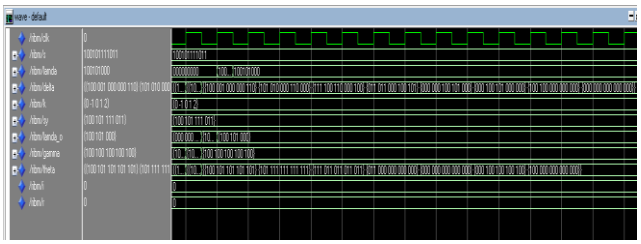

Fig. 13 RiBM output

Figure 13 shows the simulation result of RiBM block.'s' is the syndrome input. 'lamda' gives the coefficients of error locator polynomial obtained from the'delta' array. RiBM algorithm computes the error locator polynomial in 2t=4 clock cycles.

V. Conclusions

The increasing demand for the efficient and reliable digital data transmission and storage systems has led to the widespread use of Reed Solomon Forward Error Correction code. This paper intends to design a Reed Solomon encoder and LCC decoder based on unified syndrome computation. The RS encoded data after passing through the AWGN channel is reached at the decoder with different multiplicities.With Unified Syndrome Computation architecture, syndrome computation of all the test vectors is obtained in order to select the correct test vector in case of a decoding failure. Also the error polynomial is generated using KES to locate the error and its magnitude for error correction.

Acknowledgment

We would like to thank the Principal, Head of the Department and all the teaching and non-teaching staffs of TKM Institute of Technology for helping us to complete the work as mentioned in the paper.

REFERENCES

[1] Wei Zhang, Hao Wang, Boyang Pan,"Reduced Complexity LCC Reed Solomon Decoder based on Unified Syndrome Computation*", Proceedings of the IEEE transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, pp.974-978, 2013.

[2] Jaydeb Bhaumik, Sundar Das, Satyajit "Design of RS (255,251) Encoder and Decoder on FPGA",*International Journal of Soft Computing and Engineering*, volume 2, pp.2231-2307, 2013

[3] Xinmiao Zhang and Jiangli Zhu "Hardware complexities of algebraic soft decision reed solomon decoders and comparisons ," *in circuits and systems (APCCAS) 2012 IEEE Asia pacific conference* pp. 216-219, 2012

[4] F. Garcia-Herrero, J. Valls, and P. K. Meher, "High speed RS(255, 239) decoder based on LCC decoding,"*Circuits Syst. Signal Process.*,vol. 30, no. 6, pp. 1643–1669, 2011.

[5] Chang Limin, Song Lu, Duan Fengyang, "Research and Realization of RS Codec based on FPGA Technique*", Proceedings of the IEEE International Conference on Mechatronics and Automation*, pp.481-486, 2009

[6] Xinmiao Zhang, "High Speed VLSI architecture for low complexity Chase Soft- decision Reed Solomon Decoding", *Information Theory and application workshop*, 2009

[7] Chang Limin, Song Lu, Duan Fengyang, "Research and Realization of RS Codec based on FPGA Technique*", Proceedings of the IEEE International Conference on Mechatronics and Automation*, pp.481-486, 2009

[8] Shu Lin, Daniel J. Costello, "Error Control Coding: Fundamentals and Applications", Second Edition, Prentice-Hall, 2005

[9] Arshad Ahmed, Ralf Koetter, Naresh R. Shanbhag, "VLSI architectures for soft decision decoding of Reed Solomon codes ", *Proceedings of IEEE international Conference on Communications*, Vol. 5,pp.2584-2590, 2004

[10] Bernard Sklar, "*Digital Communications: Fundamentals and Applications*", Second Edition, Prentice-Hall, 2001

IJERT