A VHDL Implementation of RS-Encoder and Decoder in Spacecraft Technology with Security Enhanced

Surbhi Dubey[1] Pallavee Jaiswal[2]
[1]M.Tech Student [2]Asst. Professor
[1,2]Dept. Electronics & Communication Engineering, Dr. C.V. Raman University, Bilaspur, Chhattisgarh-India

Abstract— *In digital communication, Reed-Solomon (RS) codes refer to as a part of channel coding that had becoming very significant to better withstand the effects of various channel impairments such as noise, interference and fading. This signal processing technique is designed to improve communication performance and can be deliberate as medium for accomplishing desirable system trade-offs. The encoder attaches parity symbols to the data using a predetermined algorithm before transmission. At the decoder, the syndrome of the received codeword is calculated. VHDL implementation creates a flexible, fast method and high degree of parallelism for implementing the Reed – Solomon codes. The purpose of this paper to study the space craft encoding and decoding process using VHDL implementation of RS encoder and decoder.*

Keywords— VHDL, RS-Encode, RS-Decoder

I. INTRODUCTION

A VLSI encoder is presented that can function either as an encoder or decoder for Reed-Solomon codes. VLSI is one approach to implementing high-performance Reed-Solomon decoders. A code is the set of all the encoded words, the code word that an encoder can produce. When actual set of data encoded it becomes a code. Reed Solomon error correcting codes (RS codes) are widely used in communication systems and data storages to recover data from possible errors that occur during transmission and from disc error respectively. One typical application of the RS codes is the Forward Error Correction (FEC), the scheme is presented in Figure 1.1.

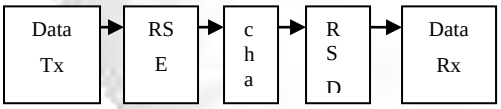


Figure 1.1: Forward Error Correction concept

Before data transmission, the encoder attaches parity symbols to the data using a predetermined algorithm before transmission. At the receiving side, the decoder detects and corrects a limited predetermined number of errors occurred during transmission. Transmitting the extra parity symbols requires extra bandwidth compared to transmitting the pure data. However, transmitting additional symbols introduced by FEC is better than retransmitting the whole package when at least an error has been detected by the receive

Reed-Solomon Code

The Reed-Solomon code was developed in 1960 by I. Reed and G. Solomon. This code is an error detection and correction scheme based on the use of Galois field arithmetic. This section provides background information on binary and extended Galois fields and summarizes the essence of the Reed-Solomon codes.

Galois Fields

A number field has the following properties:

- Both an addition and a multiplication operation that satisfy the commutative, associative, and distributive laws.
- Closure, so that adding or multiplying elements always yields field elements as results.
- Both zero and unity elements. The zero element

Here is a one-to-one correspondence between any binary number and a polynomial in that every binary number can be represented as a polynomial over GF (2), and vice versa. A polynomial of degree D over GF (2) has the following general form:

$$f(x) = f_0 + f_1x + f_2x^2 + f_3x^3 \ldots\ldots + f_D x^D \quad \ldots\ldots$$

Where the coefficients f_0..... f_D are taken from GF (2). A binary number of (N+1) bits can be represented as an abstract polynomial of degree N by taking the coefficients equal to the bits and the exponents of x equal to the bit locations. For example, the binary number 100011101 is equivalent to the following polynomial:

$$100011101 \leftrightarrow 1 + x^2 + x^3 + x^4 + x^8 \ldots\ldots\ldots\ldots\ldots\ldots$$

The bit at the zero position (the coefficient of x^0) is equal to 1, the bit at the first position (the coefficient of x) is equal to 0, and the bit at the second position (the coefficient of x^2) is equal to 1, and so on. Operations on polynomials, such as addition, subtraction, multiplication and division, are performed in an analogous way to the real number field. The sole difference is that the operations on the coefficients are performed under modulo-2 algebra. For example, the multiplication of two polynomials is as follows:

$$(1 + x^2 + x^3 + x^4).\ (x^3 + x^5) = x^3 + x^5 + x^5 + x^6 + x^7 + x^7 + x^8 + x^9 = x^3 + x^6 + x^8 + x^9$$

This result differs from the result obtained over the real number field (the middle expression) due to the XOR operation (the + operation). The terms that appear an even number of times cancel out, so the coefficients of x^5 and x^7 are not present in the end result.

Extended Galois Fields GF (2^m)

A polynomial p(x) over GF (2) is defined as irreducible if it cannot be factored into non-zero polynomials over GF (2) of smaller degrees. It is further defined as primitive if $n = (x^n + 1)$ divided by p(x) and the smallest positive integer n equals $2^m - 1$, where m is the polynomial degree. An element of GF (2^m) is defined as the root of a primitive polynomial p(x) of degree m. An element α is defined as primitive if

$$\alpha^{\,i \bmod (2m-1)}$$

where $i \in N$, can produce 2^{m-1} field elements (excluding the zero element). In general, extended Galois fields of class GF (2^m) possess 2^m elements, where m is the

symbol size, that is, the size of an element, in bits. For example, in ADSL systems, the Galois field is GF (256). It is generated by the following primitive polynomial

$1+x^2+x^3+x^4+x^8$

This is a degree-eight irreducible polynomial. The field elements are degree-seven polynomials. Due to the one-to-one mapping that exists between polynomials over GF(2) and binary numbers, the field elements are represent as binary numbers of eight bits each, that is, as bytes. In GF (2^m) fields, all elements besides the zero elements can be represented in two alternative ways: In binary form, as an ordinary binary number. In exponential form, as α^p. It follows from these definitions that the exponent p is an integer ranging from 0 to (2^m–2). Conventionally, the primitive element is chosen as 0x02, in binary representation. As for GF (2), addition over GF (2^m) is the bitwise XOR of two elements. Galois multiplication is performed in two steps: multiplying the two operands represented as polynomials and taking the remainder of the division by the primitive polynomial, all over GF (2). Alternatively, multiplication can be performed by adding the exponents of the two operands. The exponent of the product is the sum of exponents, modulo 2^m –1.

Polynomials over the Galois field are of cardinal importance in the Reed-Solomon algorithm. The mapping between bit streams and polynomials for GF (2^m) is analogous to that of GF (2). A polynomial of degree D over GF (2^m) has the most general form:

$$f(x) = f_0 + f_1x + f_2x^2 + f_3x^3 \ldots\ldots\ldots\ldots + f_Dx^D \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

where the coefficients $f_0 – f_D$ are elements of GF (2^m). A bit stream of (N+1)m bits is mapped into an abstract polynomial of degree N by setting the coefficients equal to the symbol values and the exponents of x equal to the bit locations. The Galois field is GF (256), so the bit stream is divided into symbols of eight consecutive bits each. The first symbol in the bit stream is 00000001. In exponential representation, 00000001 becomes α^0. Thus, α^0 becomes the coefficient of x^0. The second symbol is 11001100, so the coefficient of x is α^{127} and so on

Reed-Solomon Encoder and Decoder

Reed-Solomon codes are encoded and decoded within the general framework of algebraic coding theory. The main principle of algebraic coding theory is to map bit streams into abstract polynomials on which a series of mathematical operations is performed. Reed-Solomon coding is, in essence, manipulations on polynomials over GF (2^m). A block consists of information symbols and added redundant symbols. The total number of symbols is the fixed number 2^m –1. The two important code parameters are the symbol size m and the upper bound, T, on correctable symbols within a block. T also determines the code rate, since the number of information symbols within a block is the total number of symbols, minus 2T. Denoting the number of errors with unknown locations n errors and the number of errors with known locations as $\mathbf{n}_{\text{erasures}}$, the Reed-Solomon algorithm guarantees to correct a block, provided that the following is true: $\mathbf{2n_{errors} + n_{erasures} \leq 2T}$, where T is configurable. The current implementation does not deal with erasures, and this document considers only error correction.

Encoding

When the encoder receives an information sequence, it creates encoded blocks consisting of symbols each. The encoder divides the information sequence into message blocks of K ≡ N– 2T symbols. Each message block is equivalent to a message polynomial of degree K –1, denoted as m(x). In systematic encoding, the encoded block is formed by simply appending 2T redundant symbols to the end of the K-symbols long-message block, as shown in Figure 3. The redundant symbols are also called parity-check symbols.

$N = 2^m - 1$

←------- N = K+2T Block Symbols----------→

Figure 1.2. Block Structure

The redundant symbols are obtained from the redundant polynomial p(x), which is the remainder obtained by dividing $x^{2T}m(x)$ by the generator polynomial g(x):

$$p(x) = (x^{2T}m (x)) \bmod g(x)$$

where is the generator polynomial. We choose the most frequently used generating polynomial:

$$g(x) = (x + \alpha^{p1}) (x + \alpha^{p2})(x + \alpha^{p3})(x + \alpha^{p2T})$$
$$g(x) = (x + \alpha) (x + \alpha^2) (x + \alpha^3) (x + \alpha^{2T})$$

The code-word polynomial c(x) is defined as follows:

$$c(x) = x^{2T}m (x) + p (x)$$

Since in GF (2^m) algebra, plus (+), and minus (–) are the same, the code word actually equals the polynomial x^{2T} m(x) minus its remainder under division by g(x). It follows that c(x) is a multiple of g(x). Since there is a total of 2^{mK} different possible messages, there are 2^{mK} different valid code words at the encoder output. This set of 2^{mK} code words of length N is called an (N, K) block code.

Decoding

When a received block is input to the decoder for processing, the decoder first verifies whether this block appears in the dictionary of valid code words. If it does not, errors must have occurred during transmission. This part of the decoder processing is called error detection. The parameters necessary to reconstruct the original encoded block are available to the decoder. If errors are detected, the decoder attempts a reconstruction. This is called error correction. Conventionally, decoding is performed by the Petersen-Gorenstein-Zierler (PGZ) algorithm, which consists of four parts:

1. Syndromes calculation.
2. Derivation of the error-location polynomial.
3. Roots search.
4. Derivation of error values.

Applications of Reed Solomon Encoder and Decoder

Reed Solomon codes are error correcting codes that have found wide ranging applications throughout the fields of digital communication and storage. Some of which include [10]:
(i)Storage Devices (hard disks, compact disks, DVD, barcode).
(ii) Wireless Communication (mobile phones, microwave links)
(iii)Digital Television
(iv)Broadband Modems (ADSL, DSL, etc).
(v)Deep Space and Satellite Communications Networks (CCSDS).

II. Problem Identification

Existing work

In wireless, satellite communication systems reducing error rate is crucial. High bit error rates of the wireless communication system need using numerous committals to writing ways on the info transferred. Channel committal to writing for error detection and correction helps the communication system designers to cut back the consequences of a loud channel. The performance of Reed-Solomon code that's wont to decode the info stream in electronic communication. Its targeted to be applied during a forward error correction system supported network customary to boost the general performance of the system.

Limitation of Existing Work

In previous work we are transmitting data from earth station to the satellite.
(i) The satellite communication is long distance communication and there is a requirement of high speed, high bandwidth and error free transmission.
(ii) The security of the information is very important in this transmission. The Transmitter and Receiver need powerful encoder and decoder.
(iii) The RS encoder and decoder is used for error free transmission .The limitation of existing work is that there is lack of security which is to be overcome.

There different parameters like speed, bandwidth security which play very important role in error free transmission. Detection and correction codes have been widely used in the last decades; because of them it is possible to guarantee reliable transmissions, avoiding the loss of information.
It is possible to send information to so long distances as those we can find in the space and with no loss of information. Because of this, the study of the correction and detection codes has been studied all codes which have been implemented; there is one which is presented in the most applications.

III. Methodology

RS Encoder

The Reed-Solomon encoder reads in k data symbols, computes the n - k parity symbols, and appends the parity symbols to the k data symbols for a total of n symbols. The encoder is essentially a 2t tap shift register where each register is m bits wide. The multiplier coefficients are the coefficients of the RS generator polynomial. The general idea is the construction of a polynomial; the coefficients produced will be symbols such that the generator polynomial will exactly divide the data/parity polynomial.

The transmitted codeword is systematically encoded and defined in as a function of the transmitted message m(x), the generator polynomial g(x) and the number of parity symbols 2t as given below.
$c(x)=m(x) * 2t + m(x) \bmod g(x)$
Where, g(x) is the generator polynomial of degree 2t and given by,
$g(x) = (x + \alpha^{i})(x + \alpha^{i+1})\ldots(x + \alpha^{i+2t-2})(x + \alpha^{i+2t-1})$

Forming Codeword

Let a message or data unit is represented in the polynomial form as, And the codeword be represented as, This represents the result of multiplication of the data unit with the generator polynomial. One important property of G(x) is that it exactly divides c (x), assume Q(x) and P(x) to be the corresponding quotient and remainder, hence the codeword looks like, Here P(x) is the polynomial which represents the check symbols.Table 1 contains a list of some primitive polynomials.

m^a		m	
3	$1+X+X^3$	14	$1+X+X^6+X^{10}+X^{14}$
4	$1+X+X^4$	15	$1+X+X^{15}$
5	$1+X^2+X^5$	16	$1+X+X^3+X^{12}+X^{16}$
6	$1+X+X^6$	17	$1+X^3+X^{17}$
7	$1+X^3+X^7$	18	$1+X^7+X^{18}$
8	$1+X^2+X^3+X^4+X^8$	19	$1+X+X^2+X^5+X^{19}$
9	$1+X^4+X^9$	20	$1+X^3+X^{20}$
10	$1+ X^3+X^{10}$	21	$1+X^2+X^{21}$
11	$1+X^2+X^{11}$	22	$1+X+X^{22}$
12	$1+X+X^4+X^6+X^{12}$	23	$1+X^5+X^{23}$
13	$1+X+X^3+X^4+X^{13}$	24	$1+X+X^2+X^7+X^{24}$

TABLE I: SOME PRIMITIVE POLYNOMIALS

Dividing by the generator polynomial and rewriting gives,
Here, Q(x) can be identified as ratio and – P(x) as a remainder after division by G(x). The idea is that by concatenating these parity symbols to the end of data symbols, a codeword is created which is exactly divisible by g(x). So when the decoder receives the message block, it divides it with the RS generator polynomial. If the

remainder is zero, then no errors are detected, else indicates the presence of errors.

RS Encoder operation for spacecraft

RS codes are systematic, so for encoding, the information symbols in the codeword are placed as the higher power coefficients. This requires that information symbols must be shifted from power level of (n-1) down to (n-k) and the remaining positions from power (n-k-1) to 0 be filled with zeros. Therefore any RS encoder design should effectively perform the following two operations, namely division and shifting. Both operations can be easily implemented using Linear-Feedback Shift Registers.

Reed-Solomon codes may be shortened by (conceptually) making a number of data symbols zero at the encoder, not transmitting them, and then re-inserting them at the decoder. The encoder is essentially a 2t tap shift register where each register is m bits wide. The multiplier coefficients are the coefficients of the RS generator polynomial. The general idea is the construction of a polynomial; the coefficients produced will be symbols such that the generator polynomial will exactly divide the data/parity polynomial. The Encoder architecture shows that one input to each multiplier is a constant field element, which is a coefficient of the polynomial $g(x)$. For a particular block, the information polynomial $M(x)$ is given into the encoder symbol by symbol. These symbols appear at the output of the encoder after a desired latency, where control logic feeds it back through an adder to produce the related parity. This process continues until all of the k symbols of *M(x)* are input to the encoder. During this time, the control logic at the output enables only the input data path, while keeping the parity path disabled. With an output latency of about one clock cycle, the encoder outputs the last information symbol at (k+1)th clock pulse. Also, during the first k clock cycles, the feedback control logic feeds the adder output to the bus. After the last symbol has been input into the encoder (at the kth clock pulse), a wait period of at least n-k clock cycles occurs. During this waiting time, the feedback control logic disables the adder output from being fed back and supplies a constant zero symbol to the bus. Also, the output control logic disables the input data path and allows the encoder to output the parity symbols (k+2th to n+1th clock pulse). Hence, a new block can be started at the n+1th clock pulse. Encoder block diagram is shown in figure 3.1

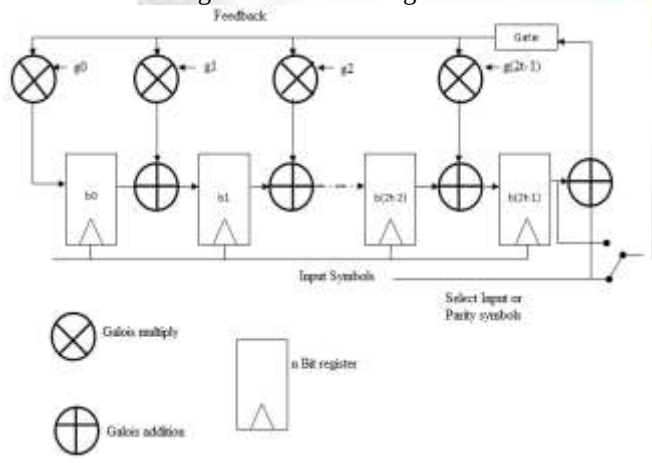


Figure 3.1: Block diagram of RS Encoder

The encoder block diagram shows that one input to each multiplier is a constant field element, which is a coefficient of the polynomial g(x). For a particular block, the information polynomial M(x) is given into the encoder symbol by symbol. These symbols appear at the output of the encoder after a desired latency, where control logic feeds it back through an adder to produce the related parity. This process continues until all of the k symbols of M(x) are input to the encoder. During this time, the control logic at the output enables only the input data path, while keeping the parity path disabled. With an output latency of about one clock cycle, the encoder outputs the last information symbol at (k+1)th clock pulse. Also, during the first k clock cycles, the feedback control logic feeds the adder output to the bus. After the last symbol has been input into the encoder (at the kth clock pulse), a wait period of at least n-k clock cycles occurs. During this waiting time, the feedback control logic disables the adder output from being fed back and supplies a constant zero symbol to the bus. Also, the output control logic disables the input data path and allows the encoder to output the parity symbols (k+2th to n+1th clock pulse). Hence, a new block can be started at the n+1th clock pulse.

RS Decoder

When a RS Decoder corrects a symbol, it replaces the incorrect symbol with the correct one, whether the error was caused by one bit being corrupted or all of the bits being corrupted. Thus, if a symbol is wrong, it might as well be wrong in all of its bit positions. This gives RS codes tremendous burst-noise advantages over binary codes. The decoding procedure for Reed- Solomon codes involves determining the locations and magnitudes of the errors in the received polynomial *R(x)*.Locations are those powers of *x* (*x2*, *x3*, and others) in the received polynomials whose coefficients are in error. Magnitudes of the errors are symbols that are added to the corrupted symbol to find the original encoded symbol. These locations and magnitudes constitute the error polynomial. Also, if the decoder is built to support erasure decoding, then the erasure polynomial has to be found. An erasure is an error with a known location. Thus, only the magnitudes of the erasures have to be found for erasure decoding. A RS (n, k) code can successfully correct as many as 2t = n-k erasures if no errors are present. With both errors and erasures present, the decoder can successfully decode if n-$k \geq 2t + e$, where t is the number of errors, and e is the number of erasures. The first step is to calculate the syndrome values from the received codeword. These are then used to find the coefficients of the error locator polynomial *Λ1 ....Λv* and the error magnitude polynomial *Ω0 ....Ωv-1* using the Euclidean algorithm . The error locations are figured out by the Chien search and the error magnitudes are calculated.The Reed-Solomon decoder tries to correct errors and 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. If there are errors present, the decoder tries to find the locations of the errors using the Berlekamp-Massey 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. For an RS (n, k) code where n - k = 2T, the decoder can correct up to T symbol

errors in the code word. Given that errors may only be corrected in units of single symbols (typically 8 data bits), Reed-Solomon coders work best for correcting burst errors. After going through a noisy transmission channel, the encoded data can be represented as r(x) = c(x) + e(x), where e(x) represents the error polynomial with the same degree as c(x) and r(x). Once the decoder evaluates e(x), the transmitted message, c(x), is then recovered by adding the received message, r(x), to the error polynomial, e(x), as given below:

c(x) = r(x) + e(x) = c(x) + e(x) + e(x) = c(x)

Note that e(x) + e(x) = 0 because addition in Galois field is equivalent to an exclusive-OR i.e e(x) XOR e(x) = 0. A typical decoder follows the following stages as shown in figure 4.2 in the decoding cycle, namely

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

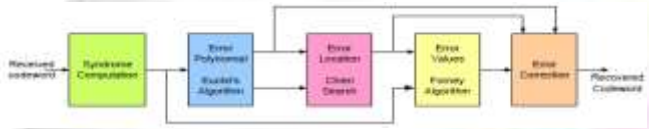
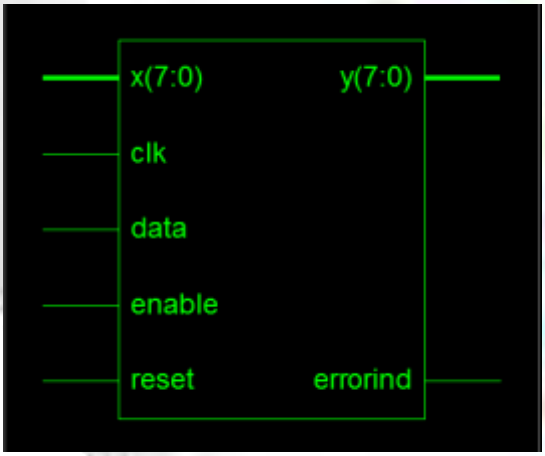
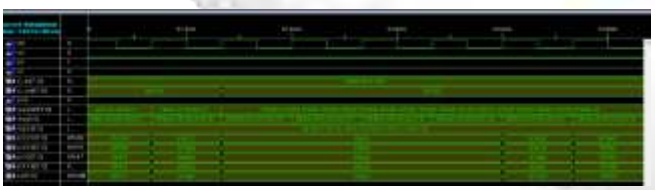


Figure 3.2: General Architecture of RS Decoder

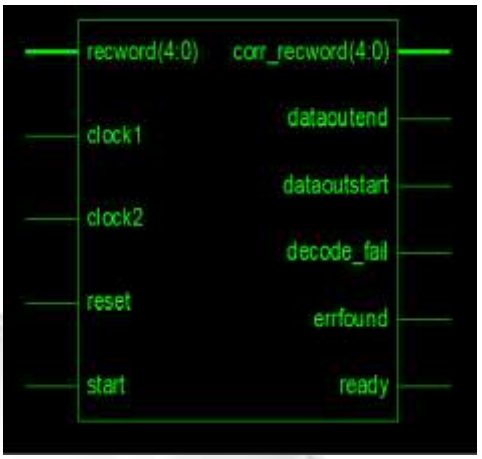
IV. Result & Discussion



4.1 RTL schematic of RS encoder



4.2 Test bench wave form of RS encoder



4.3 RTL schematic of RS decoder

V. Conclusion & Future Scope

When a RS Decoder corrects a symbol, it replaces the incorrect symbol with the correct one, whether the error was caused by one bit being corrupted or all of the bits being corrupted. The spacecraft technology needs powerful encoder and decoder for encoding and decoding with error free transmission of messages . Here Error detection and correction techniques have been used which are essential for reliable communication over a noisy channel. A compact and fast hardware implementation technique of Reed Solomon Encoding and Decoding algorithm were presented. The VHDL implementation of S encoder and Decodeer is take place. The results demonstrate that the Reed Solomon codes are very efficient for the detection and correction of burst errors.Reed Solomon codes provide a wide range of code values that can be chosen to optimize performance. RS codes are used significantly in Wireless Communication (mobile phones, microwave links), Deep Space and Satellite Communications Networks (CCSDS), mass storage devices (hard disk drives, DVD, barcodes), digital TV, digital video broadcasting (DVB), and Broadband Modems (ADSL, VDSL, SDSL, HDSL etc). Technologies are becoming smarter and compact day by day, so we hope our work will add new dimension in that trend.

Reference

[1] Ying Xhang. Implementation of RS encoder for CCSDS in IEEE Signal Processing (ICSP), 2014 12th International Conference on IEEE

[2] Aqib Al Azad and Md Imam Shahed IJFCC. A Compact and Fast FPGA Based Implementation of Encoding and Decoding Algorithm Using Reed Solomon Codes

[3] Ranjita Singh1 Krikshankant Pathak2 VHDL Implementation of Reed-Solomon Encoder-Decoder for WiMax Network

[4] I. S. Reed and G. Solomon, ― Polynomial codes over certain finite fields‖, *Journal of the Society for Industrial and Applied Mathematics*, 8:300-304, 1960.

[5] T. R. N. Rao and E. Fujiwara, Error Control Coding for Computer Systems, *Prentice-Hall, Englewood Cliffs, NJ, USA*, 1989.

[6] H.C.Chang, C.B.Shung, CY.Lee, ―A Reed- Solomon product-code (RS-PC) decoder chip for DVD applications‖, *Solid-State Circuits, IEEE Journal* , Volume: 36 Issue: 2 , Feb. 2001 Page(s): 229 –238.

[7] Y.Cho, K.Kang, J.Lee and H.Shin ―Proactive Reed-Solomon Bypasss (PRSB): A Technique for Real-Time Multimedia Processing in 3G Cellular Broadcast Networks‖, *The 11th IEEE International Conference on Embedded and Real-Time computing system and application*, 17-19 August 2005, Hong Kong

[8] J.SHIAN LI M.W. GUO, ―Improving 802.11 Wireless TCP Performance with Adaptive Reed-Solomon Codes: An Experimental Study‖, *Journal Of Information Science And Engineering* 21, 1201-1211 (2005). International Journal of Analytical, Experimental and Finite Element Analysis (IJAEFEA), Issue. 3, Vol. 1, July 2014 © 2014 RAME IJAEFEA 6 Research Association of Masters of Engineering www.rame.org.in

[9] M.F.Arnal, ―Optimisation de la fiabilité pour des communications multipoint par satellite géostationnaire, *Thèse, Ecole Nationale supèrieure des télecommunications*

[10] DVB, ―Framing structure, channel coding and modulation for digital terrestrial television,‖ *ETSI EN 300* 744, vol. 4.1, January 2001

[11] Lee H., ―A high speed, low complexity Reed-Solomon decoder for optical communications,‖ IEEE Transactions on Circuits and Systems II, PP. 461-465, 2005

[12] WiMAX Forum: ‖Mobile WiMAX. Part I: A TechnicalOverview and Performance Evaluation,‖ August 2006.

[13] Peter J. Ashenden, ―The Designer's Guide to VHDL‖, Second Edition, Morgan Kaufmann Publishers, 2004.