Implementation of Reed-Solomon Encoder and Decoder using FPGA

Mr. Swagat Karve
Department of Electronics and Telecommunication Engineering
SVERI's College of Engineering, Pandharpur, Maharashtra, India

Mr. Akshay Jadhav
Department of Electronics and Telecommunication Engineering
SVERI's College of Engineering, Pandharpur, Maharashtra, India

Mr. Pranav Chippalkatti
Department of Electronics and Telecommunication Engineering
JSPM's RSCOE, Pune, Maharashtra, India

***Abstract* — In the present world, communication has got many applications such as telephonic conversations etc. in which the messages are encoded into the communication channel and then decoding it at the receiver end. During the transfer of message, the data might get corrupted due to lots of disturbances in the communication channel. So it is necessary for the decoder tool to also have a function of correcting the error that might occur. Reed Solomon codes are type of burst error detecting codes which has got many applications due to its burst error detection and correction nature. My aim of the project is to implement this reed Solomon codes in a VHDL test bench waveform and also to analyses the error probability that is occurring during transmission. To perform this check one can start with simulating Reed Solomon codes in MATLAB and then going for simulation in XILINX writing the VHDL code [4]. The encoder and decoder design of reed Solomon codes have got different algorithms. Based on your requirements you can use those algorithms. The difference between the algorithms is that of the computational calculations between them. The complexity of the code depends on the algorithm used. I will be using Linear Feedback Shift Register circuit for designing the encoder.**

***General Terms* - Berlekamp-Massey algorithm, Chien-search algorithm, Forney algorithm, Digital Communication, Channel Coding.**

***Key Words* - Reed Solomon Coding, Feed Back Shift Register, Encoder, Decoder, Parity, Coding gain.**

I. INTRODUCTION

Communication Engineering is been the vital field of engineering in last few decades Evolution of digital communication has made this field more interesting as well as challenging. All the advancements in the field of communication are to achieve two important goals namely reliability and efficiency. In most cases reliability is given the priority over efficiency though at certain cases one is compromised for the other. Reliability of communication has an impact even in our day to day life. For example message received on our mobile phone may become unreadable if some error occurs during the transmission, or a scratch in our DVD may make it unreadable. There are wide ranges of concern in the field of digital communication. Error control issues have been addressed in this Paper.

A. Errors and Methods

Generally communication is understood as transmission and reception of data from one place to other at some distance. If we change the reference it can also include transmission and reception of data at the same place but at a different point of time, which means storage and retrieval of data.

Hence storage is also a part of communication. In any system application we come across errors either in communication or in storage. Errors in transmission are mainly because of noise, electromagnetic interferences, crosstalk, bandwidth limitation, etc. In case of storage, errors may occur because of increase in magnetic flux as in case of magnetic disc or it can be spurious change of bits because of electromagnetic interferences as in case of DRAM. Hence dealing with these errors when they occur is the matter of concern. The first step is to detect the error. And after the error gets detected there are two alternate approaches to proceed.

a) Automatic repeat request (ARQ): In this approach, the receiver first detects the error and then sends a signal to the transmitter to retransmit the signal. This can be done in two ways (i) continuous transmission mode (ii) wait for acknowledgement. In continuous transmission mode, the data is being sent by the transmitter continuously. Whenever receiver finds any error it sends a request for retransmission. However, the retransmission can either be selective repeat or go back N step type. As the name suggests, in selective repeat those data units containing error are only retransmitted. While in go back N type, retransmission of last N data unit occurs. Next, in wait for acknowledgement mode, acknowledgement is sent by the receiver after it correctly receives each message. Hence, when not sent, the retransmission is initiated by the transmitter.[6]

b) Forward error correction (FEC): In this approach, error is both detected and corrected at the receiver end. To enable the receiver to detect and correct the data, some redundant information is sent with the actual information by the transmitter .After being introduced to both the approaches, one should choose whether which approach is to be used. Automatic repeat request is easier but if the error occurs much frequently, then retransmission at that frequency will particularly reduce the effective rate of data transmission. However, in some cases retransmission may not be feasible to us. In those cases, Forward Error correction would be more suitable. As Forward Error Correction involves additional information during transmission along with the actual data. It also reduces the effective data rate which is independent of rate of error. Hence, if error occurs less frequently then Automatic request approach is followed keeping in mind that retransmission is feasible. Out of the various FEC's, Reed Solomon code is one. These are block error correcting codes with wide range of applications in the field of digital communications. These codes are used to correct errors in devices such as CD, DVD, wireless Communication etc [2][5].

B. Scope of the Work

As stated in the last paragraph, RS codes are used for many applications. With the objective of developing high speed RS codes, the scope includes: Study of different error detection and correction methods, implementation of RS encoder and decoder in Hardware Description Language (HDL), building fully synchronous synthesizable logic core of RS encoder and decoder to meet the requirement of almost all the standards that employ RS codes, such as CCSDS, DVB, ETIS-BRAN, IEEE802.6, G.709, IESS-308, etc.[1][2][5].

II. REED SOLOMON CODES

Reed Solomon codes are non binary cyclic error correcting codes. They describe a systematic way of building codes that can detect and correct multiple errors. In a block code we have k individual information bits, r individual parity bits and a total of n (= k + r) bits. Rather reed Solomon codes are organized in group of bits. These group of bits are referred to as symbols. So we can say this code has n number of symbols. This symbol comprises of m number of bits where the n is

$$n=2^m-1$$

These n symbols form a code word. Out of these n symbols k are information symbols where n-k are parity symbols. These codes are used to transfer data between any two medium. There are possibilities that error may occur due to any disturbance in the traverse channel. So we need to have a design tool specified for encoding and then decoding after correcting signals in the receiver side. Reed Solomon codes can be the tool discussed above. The error correcting capability of the code is defined as t, where [6]

$$t=[(n-k)/2]$$

where [x] represents the greater integer function. The code can correct only half of the number of parity symbols because for every error one parity symbol is used to locate the error and the other is used to correct it. The advantage behind reed Solomon code is that it can correct burst errors. However if erasures are present then only one parity symbol is used to correct error as the location of error is already known. The code rate is defined as R_C, where

$$R_C = k/n$$

A. Properties of Reed Solomon Codes

The properties of Reed-Solomon codes make them especially well-suited to applications where errors occur in bursts. This is because [4][6].

i) It does not matter to the code how many bits in a symbol are in error—if multiple bits in a symbol are corrupted it only counts as a single error. Alternatively, if a data stream is not characterized by error bursts or drop-outs but by random single bit errors, a Reed-Solomon code is usually a poor choice. More effective codes are available for this case.

ii) Designers are not required to use the "natural" sizes of Reed-Solomon code blocks. A technique known as "shortening" can produce a smaller code of any desired size from a larger code. For example, the widely used (255,251) code can be converted to a (160,128) and not transmitting them. At the decoder, the same portion of the block is loaded locally with binary zeroes.

iii) Its capability to correct both burst errors and erasures makes it the best choice for the designer to use it as the encoding and decoding tool.

III. ARCHITECTURE OF RS CODES

A. Encoder Architecture

Reed-Solomon encoding and decoding can be carried out in software or in special-purpose hardware.

1) Finite (Galois) Field Arithmetic:

Reed-Solomon codes are based on a specialist area of mathematics known as Galois fields or finite fields. A finite field has the property that arithmetic operations (+,-,x,/ etc.) on field elements always have a result in the field. A Reed Solomon encoder or decoder needs to carry out these arithmetic operations. These operations require special hardware or software functions to implementation Generator Polynomial. [6][4]

2) Generator Polynomial:

A Reed-Solomon codeword is generated using a special polynomial. All valid codewords are exactly divisible by the generator polynomial. The general form of the generator polynomial is

$$g(x)=(x-\alpha^{i})(x-\alpha^{i+1})\ldots\ldots(x-\alpha^{i+2i}) \quad \text{Eq. 1}$$

and the codeword is constructed using

$$c(x) = g(x).i(x) \quad \text{Eq. 2}$$

where g(x) is the generator polynomial, i(x) is the information block, c(x) is a valid codeword and a is referred to as a primitive element of the field .

3) Parity Generator:

The *2t* parity symbols in a systematic Reed-Solomon codeword are given by

$$p(x)= i(x)*X^{n-k} \bmod g(x) \quad \text{Eq.3}$$

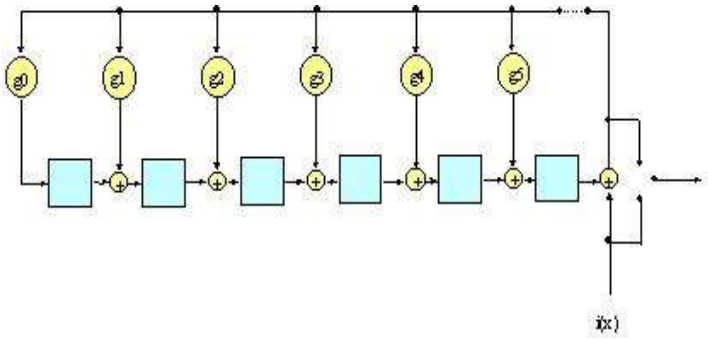

Figure 1 Reed Solomon Encoder

B. Design Hierarchy of Encoder Block

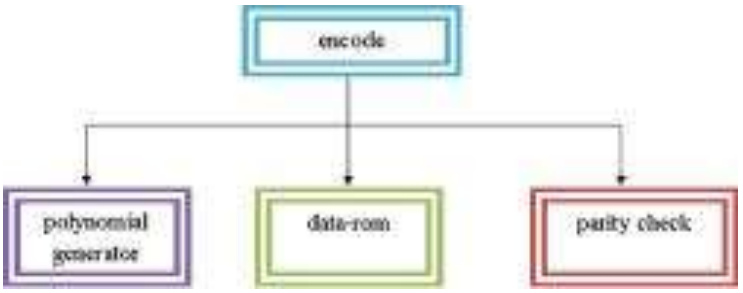

Figure 2 Design hierarchies for RS Encoder

C. Encoder Operation

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 i(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 i(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 k^{th} 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 ($k+2^{th}$ to $n+1^{th}$ clock pulse). Hence, a new block can be started at the $n+1^{th}$ clock pulse.[4].

D. Decoder Architecture

A Reed-Solomon decoder attempts to identify the position and magnitude of up to t errors (or 2t erasures) and to correct the errors or erasures.

1) Syndrome Calculation:

This is a similar calculation to parity calculation. A Reed Solomon codeword has 2t syndromes that depend only on errors (not on the transmitted code word). The syndromes can be calculated by substituting the 2t roots of the generator polynomial g(x) into r(x).

2) Finding the Symbol error Location:

This involves solving simultaneous equations with t unknowns. Several fast algorithms are available to do this. These algorithms take advantage of the special matrix structure of Reed-Solomon codes and greatly reduce the computational effort required. In general two steps are:

a) Find an error locator polynomial: This can be done using the Berlekamp-Massey algorithm or Euclid's algorithm. Euclid's algorithm tends to be more widely used in practice because it is easier to implement: however, the Berlekamp-Massey algorithm tends to lead to more efficient hardware and software implementations.

b) Find the roots of this Polynomial:

This is done using the Chien-search algorithm.

3) Finding the symbol error values:

Again, this involves solving simultaneous equations with t unknowns. A widely-used fast algorithm is the Forney algorithm [6].

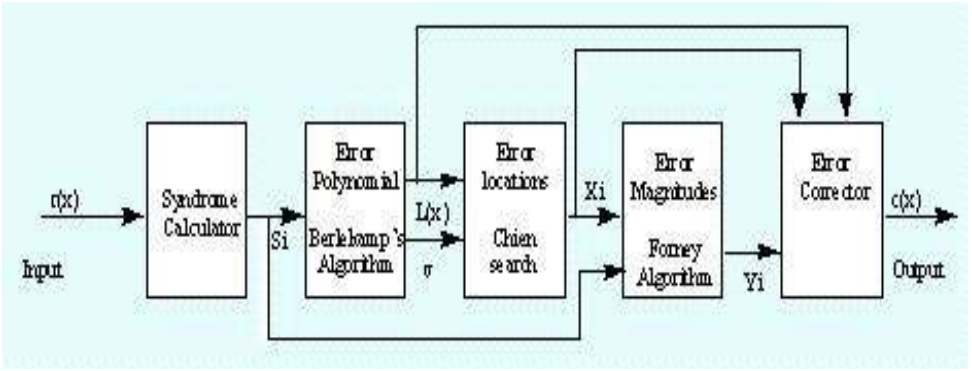

Figure 3 Reed Solomon Decoder

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

The received codeword r(x) is the original (transmitted) codeword c(x) plus errors introduced during transmission.

$$r(x) = c(x) + e(x) \qquad \text{Eq. 4}$$

E. Design Hierarchy of Decoder Block

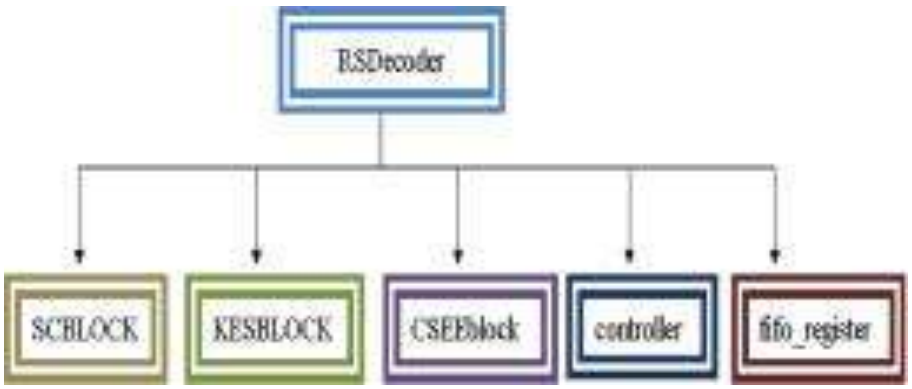

Figure 4 Design hierarchies for RS Decoder

F. Decoder Operation

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 .

RS Decoder mainly works on five steps:

1) Calculate the syndromes.
2) Use the syndromes to determine the "error locator polynomial."
3) Find the roots of the error locator polynomial. The inverses of these roots give the locations of errors.
4) Use the syndromes, roots, and error locator

Polynomial to determine the error magnitudes.

5) Use the information about error location and magnitude to actually correct the errors.[4]

IV. HARDWARE AND SOFTWARE IMPLEMENTATION OF RS CODES

A. Hardware Implementation

RS decoder design can also be implemented on the Field Programmable Gate Array (FPGA). This device is similar to the gate array, defined above, with the device shipped to the user with general-purpose metallization pre-fabricated, often with variable length segments or routing tracks. The device is programmed by turning on switches which make connections between circuit nodes and the metal routing tracks. The connection may be made by a transistor switch (which are controlled by a programmable memory element) or by an antifuse. The transistor switch may be controlled by an SRAM cell or an EPROM/EEPROM/Flash cell. Timing is generally not easily predictable. Some architectures employ dedicated logic and routing resources for optimizing high-speed functions such as carry chains, wide decodes, etc. Additional features such as programmable I/O blocks, storage registers, etc., may be included in these devices. Commercial, military, and space devices use a variety of programmable elements. General Purpose Processors (GPPs), Digital Signal Processors (DSPs), or hybrid ASICs can provide flexibility, but do so at the expense of processing time and area. FPGAs, however, promise to realize RS decoders with time and area performance similar to ASICs but with flexibility similar to GPPs. This is intriguing as it opens the door to dynamically changing error control coding based on the current characteristics of the transmission channel. Indeed, FPGA hosted RS decoders already exist but little work has been reported on realizing FPGA-based dynamic error control or on simply characterizing the optimal FPGA implementation of RS decoders. The finite field multipliers are the most numerous building blocks of RS decoders. There are a number a RS decoder design parameters which can be adjusted to reduce the resource requirement, increase speed, and make FPGA-based finite field decoders more feasible. Nevertheless, attempting to optimize the FPGA-based finite field multipliers in terms of both speed and area is an important step. The dependence of RS decoders on finite field arithmetic implies that the efficient RS decoders are dependent on efficient finite field adders and multipliers.

Addition is easy because of the finite fields. It equates to a bit-wise XOR of the m-tuple and is realized by an array of m-XOR gates. The finite field multiplier is, by comparison, much more complicated and is the key to developing efficient finite field computational circuits. Multipliers can be classified into bit parallel and bit serial architectures. The bit parallel multipliers compute a result in one clock cycle but have generally a higher area requirement. The Bit serial multipliers compute a product in k clock cycles but have a lower area requirement. In this paper we are using SPARTON3S400 FPGA.

B. Software Implementation

Until recently, software implementations in "real-time" required too much computational power for all but the simplest of Reed-Solomon codes (i.e. codes with small values of t). The major difficulty in implementing Reed-Solomon codes in software is that general purpose processors do not support Galois field arithmetic operations. For example, to implement a Galois field multiply in software requires a test for 0, two log table look-ups, modulo add and anti-log table look-up. However, careful design together with increases in processor performance means that software implementations can operate at relatively high data rates.

C. VHDL Language

There are many reasons for choosing VHDL for design efforts. The most important feature of VHDL is that it improves productivity when used it is used in a structured, top-down approach to design. Real increases in productivity come later; when VHDL has been learnt to a good level and the user have accumulated a library of reusable VHDL components. Productivity also increases when VHDL is used to enhance communication between team members and when an advantage of the more powerful tools for simulation and design verification that are available is taken. In addition, VHDL allows to design at a more abstract level. Instead of focusing on a gate-level implementation, the behavioral function of the design can be addressed. It is easy to build and use libraries of commonly used VHDL modules. A design can be reused as many times as required in VHDL. Due to the benefits of reusable code, VHDL statements are generally written in ways that make them general purpose. Writing portable code becomes an automatic reflex. Another important reason to use VHDL is the rapid pace of development in Electronic Design Automation (EDA) tools and in target technologies. Using a standard language such as VHDL greatly improves the chances of moving into more advanced tools (for example, from a basic low-cost simulator to a more advanced one) without having to re – enter the circuit descriptions. The ability to retarget circuits to new types of device targets (for example, ASICs, FPGAs, and complex PLDs) also improve by using a standard design entry method. In this paper we are using VHDL as programming language also the results can be proposed in the advanced software such as MAT Lab.

V. PROPOSD RESULTS

As per the property of RS code the codes, as with any block code used in a systematic context, can be "shortened" to an arbitrary extent. RS codes defined over the field GF (256) can be said to have a natural length of 256 (the original approach) or 255 (the generator polynomial approach). The compact disc system is able to use RS codes of length 32 and 28 symbols, while still retaining the 8-bit symbol structure of a length-255 code. The shorter code word length keeps the implementation of the interleaver simple, while the 8-bit symbols provide protection against error bursts and provide a good match for the 16-bit samples taken from the analog music source. In this paper, a (32, 28) RS code has been implemented in VHDL. The encoding and decoding of the (32, 28) RS code with their simulation results are given here.

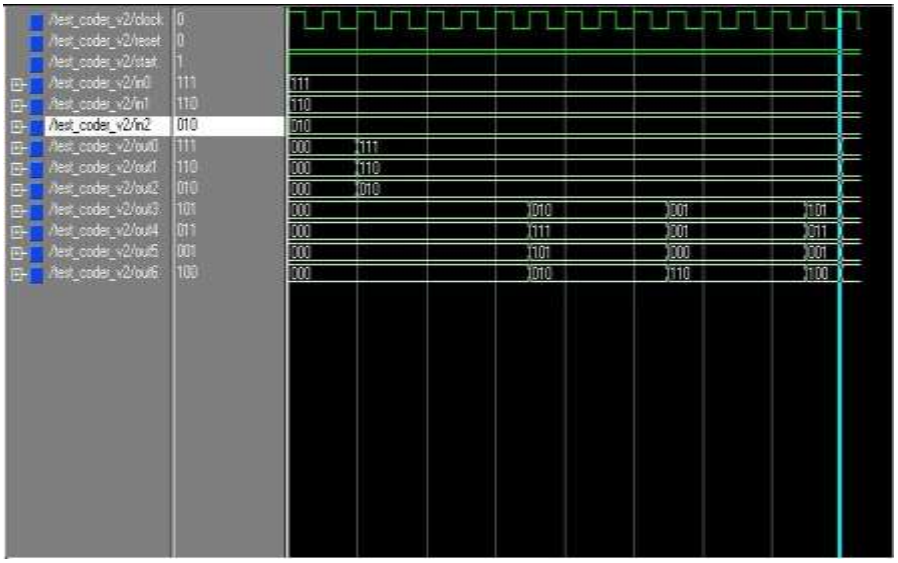

Figure 5 Simulation result of Encoder

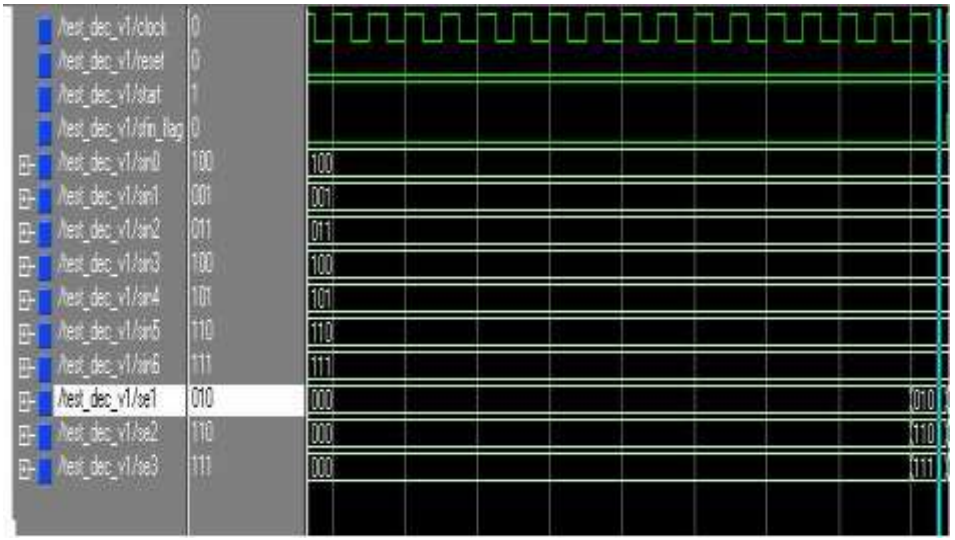

Figure 6 Simulation result of Decoder

VI. CONCLUSION AND FUTURE SCOPE

Reed – Solomon codes are one of the most powerful and efficient non-binary error correcting codes for detecting and correcting burst errors. An irreducible generator polynomial is used for generating the encoded data called codeword. All encoded data symbols are elements of the Galois field defined by the parameters of the application and the properties of the system. The encoder is implemented using linear shift registers with feedback. The decoder checks the received data for any errors by calculating the syndrome of the codeword. If an error is detected, the process of correction begins by locating the errors first. Generally Euclid's Algorithm is used to calculate the error – locator polynomial it is very easy to implement, while its counterpart Berlekamp-Massey Algorithm is more hardware efficient. The precise location of the errors is calculated by using Chien-search algorithm. Then magnitude of the errors is calculated using Forney algorithm. The magnitude of the error is added to the received codeword to obtain a correct codeword. Hence the using the Reed Solomon codes, burst errors can be effectively corrected.

Reed Solomon codes are efficiently used for compact discs to correct the bursts which might occur due to scratches or fingerprints on the discs. Convolutional encoding with Viterbi decoding has been in use for many years in commercial satellite systems. A characteristic of the Viterbi decoding process is that as the power density ratio (E / N_0) becomes increasingly smaller, the uncorrectable errors that are passed through the system are clumped together. Although the distribution of errors caused on the link is Gaussian, the uncorrectable error distribution at the Viterbi decoder output tends to occur in bursts. The obvious conclusion is to concatenate the Viterbi and Reed - Solomon codes. The resulting scheme, called Reed-Solomon / Viterbi concatenated coding offers high reliability and modest complexity. These two desirable features make Reed-Solomon codes an apparent choice for deep space probes. They have been efficiently used in the image communication system of NASA's Voyager mission. Reed-Solomon codes will continue to be used to force communication system performance ever closer to the line drawn by Shannon.

VII. ACKNOWLDGEMENT

At first the authors would want to thank entire management of SCI Team for giving the opportunity to do this work. I am thankful to all the helping hands that extended the preparatory steps of this paper. I am very much thankful to our Research Head for his support and providing all facilities to complete the report.

REFERENCES

[1] Joel Trubuil, Andre Goalic and Nicolas Beuzelin, IEEE 2013 An Overview of Channel Coding for Underwater Acoustic Communication.

[2] Bhawna Tiwari and Rajesh Mehra, IEEE 2012 Design and Implementation of Reed Solomon Decoder for 802.16 Network using FPGA.

[3] Kaikai Liu, Tian Ban, Lirida Naviner, Jean-Francois Naviner, IEEE 2012 Reliability Analysis of a Reed Solomon Decoder.

[4] Aqib. Al Azad, Minhazul. Huq, Iqbalur. Rahman Rokon, ICAEPE 2011, Efficient Hardware Implementation of Reed Solomon Encoder using verilog.

[5] Syed Abdul Baqi Sha , Saeid Nooshabadi, and Dong Soo Har, IEEE Symposium 2012, Efficient Implementation of Channel Coding and Interleaver for Digital Video Broadcasting (DVB-T2) on FPGA.

[6] B. Sklar, Prentice-hall 2001 Digital Communications: Fundamentals and Applications.

[7] Martyn Riley and Iain Richardson, Prentice-hall 2001, An introduction to Reed Solomon codes: principles, architecture & implementation.

[8] Wicker and Bhargava, Prentice-hall, An introduction to the Reed Solomon Codes.