Open access Journal International Journal of Emerging Trends in Science and Technology

Design and Implementation of High Performance Reed Solomon Encoder and Decoder

Authors

Harshada L. Borkar[1], Prof. V.N.Bhonge[2]

[1]M.E. Student, Electronics and Telecommunication, Shri Sant Gajanan Maharaj College of Engineering, Maharashtra, India
Email: *harshadaborkar1@gmail.com*
[2]Associate Professor, Electronics and Telecommunication, Shri Sant Gajanan Maharaj College of Engineering, Maharashtra, India
Email: *vnbhonge@rediffmail.com*

Abstract
In this paper, Reed Solomon (RS) Encoder and Decoder and their implementation in Spartan 6 Field Programmable Gate Array (FPGA) is analyzed. RS codes are non-binary cyclic error correcting block codes. Here parity symbols are generated at the encoder end using a generator polynomial and added to the very end of the message symbols. Then the locations and magnitudes of errors in the received polynomial are determined by the RS decoder. The main objective of this project is to optimize the area used on FPGA which in turn minimizes the size and ultimately the cost. The paper covers the RS encoding and decoding algorithm, simulations and the implementation details of the encoder and decoder architecture. Register transfer level (RTL) of RS encoder and decoder is designed, simulated and implemented using Xilinx in Spartan 6 FPGA kit.
Keywords: *FPGA, Key Equation Solver (KES), Reed Solomon (RS) Decoder, Reed Solomon (RS) Encoder and VHDL.*

1. Introduction

Communication is an important part of our daily life. New technologies are being invented every now and then to make communication easier. Fast data transmission is the need of today's world. We constantly need to increase the flow of transmission while maintaining and improving their quality. But there is a possibility of our data which we transmit may get corrupted due to some errors. So some error correcting codes must be used for correcting one or several errors in a code word by adding redundant symbols to the information, which are also called as, parity symbols.

Many digital signaling application use Forward Error Correction, a technique in which redundant information is added to the message that is transmitted so that receiver should be able to detect and correct errors that might have occurred during transmission. There are many different types of codes used for this purpose but RS codes have proved to be the most efficient among all in case of efficiency and complexity.

1.1 RS Codes

Error correcting codes are classified into two categories, the main ones being block codes and convolutional codes. A Reed-Solomon codes comes in the category of block code, which means that the message which is being transmitted is divided into separate blocks of data. Each block then has parity bit added to it to form a self-contained code word. It is also called as a systematic code, which means that the encoding process does not alter the message

symbols and the parity symbols are added as a separate part of the block. This is shown diagrammatically in Figure 1.

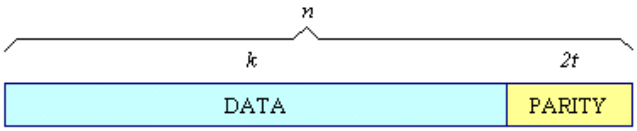

Fig -1: Reed Solomon Codeword

Also, a Reed-Solomon code is a linear code which adds two code words and produces another code words and it is also a cyclic code which shifts the symbols cyclically and produce another code word. It belongs to the family of Bose-Chaudhuri-Hocquenghem (BCH) codes, but is differentiated from BCH codes by having multi-bit symbols. This makes the code good at dealing with bursts errors because, although a symbol may have all its bits in error, this counts as only one symbol error in terms of the correction capacity of the code.

Choosing different parameters for a code provides different levels of protection and affects the complexity of implementation. Thus a Reed-Solomon code can be described as an (*n*, *k*) code, where *n* is the block length in symbols and *k* is the number of information symbols in the message. Also,

$$n = 2^m-1 \quad .... (1)$$

where *m* is the number of bits in a symbol. There are *n-k* parity symbols and *t* symbol errors can be corrected in a block, where *t* = (*n-k*)/2 for *n-k* even or *t* = (*n-k*-1)/2 for *n-k* odd. In this project RS(255,239) scheme is used in which, n = 255, k = 239 and number of parity symbols are 255 - 239 = 16 and its error correcting capability is upto 8 symbols. The specified code generator polynomials are given by:

- *Code Generator Polynomial:*

$$g(x) = (x + \mu^1)(x + \mu^2) \dots . (x + \mu^{2T})$$

Where μ=02hex

-Field Generator Polynomial:

$$p(x) = x^8+x^4+x^3+x^2+1$$

The symbol width (m) gives the field generator polynomial. Table 1 illustrates this statement.

Table -1: Relation between symbol width and the field generator polynomial

Symbol Width	Field generator polynomial
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$\mathbf{x^8+x^4+x^3+x^2+1}$
9	x^9+x^4+1
10	$x^{10}+x^3+1$
11	$x^{11}+x^2+1$
12	$x^{12}+x^6+x^4+x+1$

Since all new generation standards use eight value as symbol width this leads to use $x^8+x^4+x^3+x^2+1$ as a field generator polynomial.

2. REED SOLOMON ENCODER

First of all RS encoder needs to be designed. At the encoder side RS codes operate on the information by splitting the message stream into blocks of data. The Reed Solomon Encoder reads k data symbols, computes the n–k symbols, attach the parity symbols to the k data symbols and hence forms the code word which is also called as code vector V(x). Two main operations are performed by RS encoder i.e. shifting and division. Both operations are performed by using linear-feedback shift registers. Let a polynomial form of message symbol is represented as:

$$M(x) = m_{k-1}\, x^{k-1} + m_{k-2}\, x^{k-2} + \dots\dots\dots\dots m_1 x + m_0$$

And the generator polynomial is given below:

$$g(x) = g_0 + g_1 X + g_2 X^2 + \dots\dots\dots + g_{2t-1} X^{2t-1} + X^{2t}$$

Parity-check is given by:

$$CK(x) = x^{n-k} M(x) \bmod g(x)$$

And codeword is given by:

$$C(x) = x^{n-k} M(x) + CK(x)$$

The parity symbols are calculated by performing a polynomial division using GF algebra. First we have to multiply the message symbols by x^{n-k}. This shifts the message symbols to the left side to make place for the parity symbols. Then we have to divide the message polynomial by the code generator polynomial using GF algebra. The parity symbols are the remainder of this division. These steps are carried out in hardware using a shift register with feedback. The architecture for the encoder is shown in Figure 2 below:

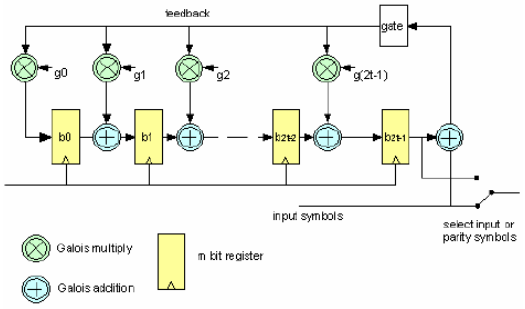

Fig -2: Architecture of RS Encoder

3. REED SOLOMON DECODER

The second step is to construct the RS decoder. At the decoder, to find the coefficients of detected errors the syndrome of the received codeword is calculated using the generator polynomial. The transmission channel, especially in space, submarine, nuclear introduces a huge amount of noise into the information message. Thus the input codeword is received at the receiver end as codeword, c(x), plus any noise or error, e(x), say r(x) = c(x) + e(x).Then to correct these errors, it is important to locate them which can be done through an error locator polynomial and using error magnitude polynomial its value is calculated. Therefore at last a correct codeword is obtained. When a RS Decoder corrects a symbol, it replaces the incorrect symbol with the correct one, whether the error was occurred in one bit or all of the bits.

The general decoding steps are illustrated in Figure 3. The syndrome calculator produces a set of syndromes from the received codeword polynomial R(x). From the syndrome values, the key equation solver computes the error locator polynomial σ(x) and the error evaluator polynomial Ω(*x*) which can be used by the Chien Search and the Error Value Evaluator to determine the error locations and error values, respectively. The block diagram for RS decoder is shown in Figure 3.

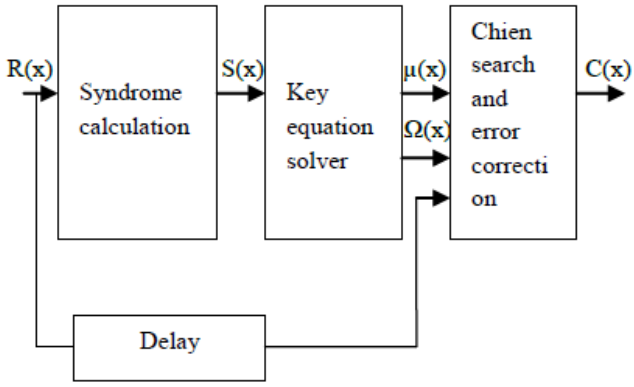

Fig -3: Block diagram of RS Decoder

3.1. SYNDROME CALCULATION:

The syndrome computation is the first block in decoding process. At the decoder side syndromes are calculated to find out error locations and from error locations error values are computed. The syndromes are nothing but the remainders obtained by dividing the received polynomial with the generator polynomial.

$$\frac{R(x)}{g(x)} = P(x) + \frac{S(x)}{g(x)} \quad - (2)$$

where, P(x) is a quotient and S(x) is the syndrome polynomial and g(x) is the generator polynomial. If E(x) is the error caused by the channel, then

$$R(x) = V(x)+E(x) \quad - (3)$$

where V(x) is the transmitted codeword and

$$\frac{R(x)}{g(x)} = \frac{V(x)}{g(x)} + \frac{E(x)}{g(x)} \quad - (4)$$

Put V(x) = g(x)*D(x) where D(x) is a message polynomial. Therefore from equation (2) and (4) we have,

$$P(x) + \frac{S(x)}{g(x)} = \frac{V(x)}{g(x)} + \frac{E(x)}{g(x)}$$

$$P(x) + \frac{S(x)}{g(x)} = \frac{g(x)*D(x)}{g(x)} + \frac{E(x)}{g(x)}$$

$$\frac{E(x)}{g(x)} = P(x) + D(x) + \frac{S(x)}{g(x)}$$

$$E(x) = [P(x) + D(x)]g(x) + S(x)$$

Therefore syndrome of R(x) i.e. the received codeword is equal to remainder resulting from dividing the error pattern by generator polynomial and the syndrome contains information about error pattern that can be used for error correction. Figure 4 properly illustrates this statement.

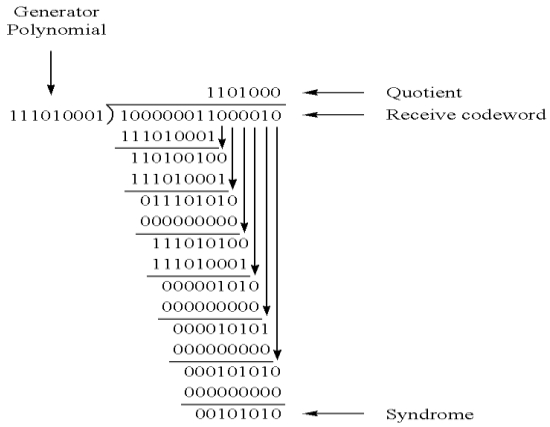

Fig -4: Example showing syndrome values

3.2. KEY EQUATION SOLVER:

The main part of RS-decoder is the key equation solver block. It solves a set of linearly dependent equations. It generates the key equations (σ(x): locator polynomial and Ω(x): evaluator polynomial) from the syndrome polynomials. The locator polynomial contains location of error symbols in the codeword. The evaluator polynomial contains information about the error magnitude of the bad symbols. The two polynomials σ(x) and Ω(x) are defined respectively by the following equations -

$$\sigma(x) = \prod_{i=1}^{w}(1 - xX_i)$$

$$\Omega(x) = \sum_{i=1}^{W} YiXi \prod_{j=1, j\neq i}^{W}(1 - Xj * x)$$

The two polynomials are related to S(x) through a Key equation that can determine the two unknown polynomials σ(x) and Ω(x) by solving the key equation given below:

$S(x)*\sigma(x) = \Omega(x) \bmod x^{2t}$

The algorithms which are mostly used to solve this key equation are the Berlekamp–Massey algorithm and the Euclidean algorithm. On comparing the Berlekamp–Massey algorithm with the Euclidean algorithm, we come to know that BM algorithm has less hardware complexity. Also it works faster than the Euclidean algorithm.

3.2.1. BERLEKAMP MASSEY ALGORITHM :

BM stands for Berlekamp-Massey algorithm and it is used for solving key equations. It is the fastest and hence often preferred algorithm. The technique used in this project is the BM algorithm because it has less hardware complexity where as the euclidean algorithm is very time consuming. In BM algorithm we just need to construct the LFSR which is hardware efficient whereas in Euclid's algorithm there are many iterations and matrix multiplications are carried out. BM algorithm solves,

$$\sum_{W}^{2t-1} \sigma(x)S(x) = 0$$

where w≤t is the number of errors that have occurred. The algorithm aims to find an LFSR of minimal length such that the first (2t) elements in the LFSR output sequence are the (2t) syndromes. The taps of this shift register are the coefficients of the desired error locator polynomial, σ(x). If one knows the syndrome values, we can compute σ(x) polynomial by the flow chart given in figure 5.

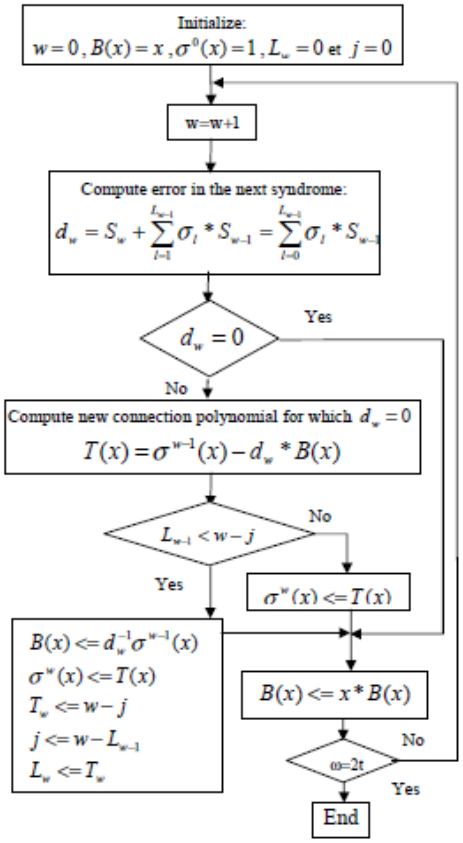

Fig -5: Flow Chart for BM algorithm

3.3. SEARCH AND ERROR EVALUATOR BLOCK:

If the error locator polynomial is known it is possible to determine the error locations by checking whether the error locator polynomial equals zero or not. The inverse of roots of the error locator polynomial are the error locations of the codeword. To find the roots of the polynomial, a Chien Search (CS) method is used. It uses all possible input values and then checks if the outputs are zeroes or not. This happens only when an error occurs. For each element that is substituted in the polynomial that equates to zero is stored into memory, as these elements are the roots of the polynomial and hence, the inverse error locations.

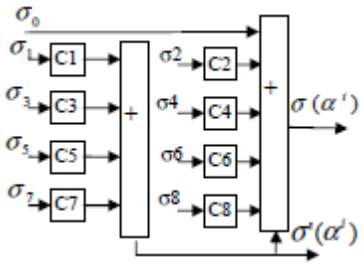

Fig -6: Chein search for t = 8

3.4. FORNEY ALGORITHM:

Once the errors are located, the next step is to use the syndrome values and the error polynomial roots to derive the error values. Forney method is generally used for this purpose. It is an efficient way of performing a matrix inversion. First the error evaluator polynomial ω(x) is calculated. This is done by convolving the syndromes with the error polynomial σ (x) as shown in this equation -

$$\omega(x) = S(x)\sigma(x) mod(x^P)$$

This calculation is carried out at each zero location, which gives the error symbol at the corresponding location. The error magnitude at each error location is given by

$$e_i = \frac{\omega(\alpha^i)}{\sigma'(\alpha^i)}$$

If the error symbol has any set bit, it means that the corresponding bit in the received symbol is at error, and must be inverted.

4. PROJECT FLOW

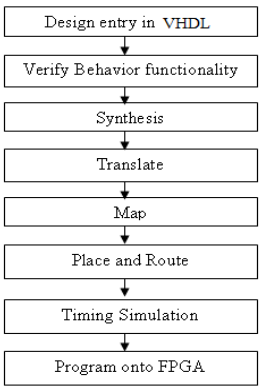

Fig -6: Program Flow

5. SIMULATION RESULTS

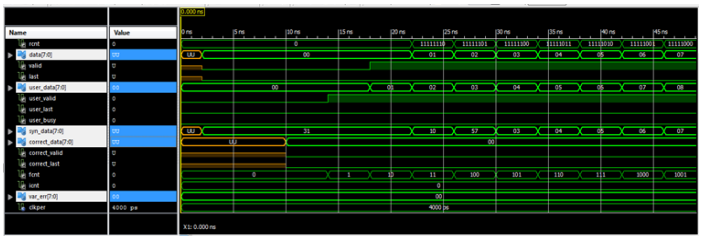

Fig -7: Output of RS Decoder

This fig. 7 shows the user data as 00, 01, 02 and so on, the encoded data i.e. data signal, syndrome data i.e. syn_data signal and corrected data is yet zero here because not all encoded data is fed to the encoder.

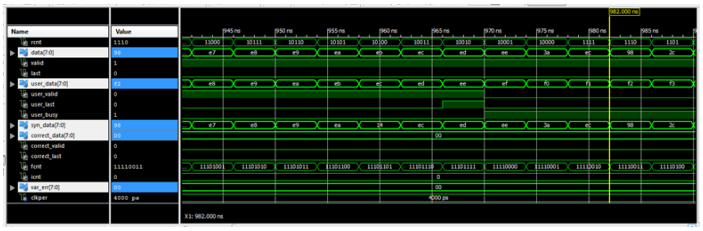

Fig -8: Output of RS decoder

This fig.8 shows an error occurred in one of the symbol. At one place the data signal shows ‘eb’ data and syn_data shows ‘14’ data, since both of them differ this indicates error.

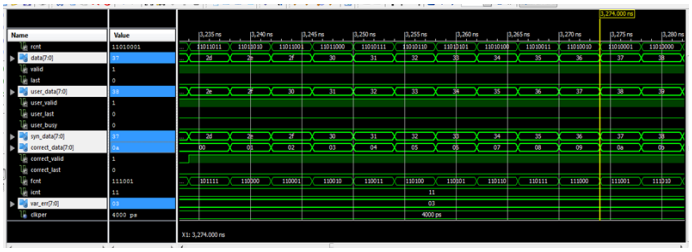

Fig -9: Output of RS decoder

This fig.9 shows the corrected data occurring at the decoder output after 255 cycles.

6. RTL DIAGRAM:

RTL stands for register transfer level. It gives a black box view of our design circuit.

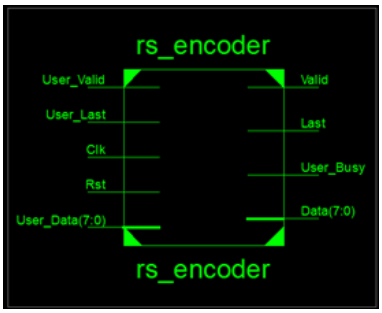

Fig -10: RTL diagram for encoder

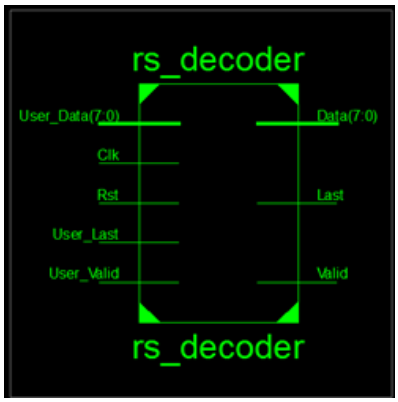

Fig -11: RTL diagram for decoder

Table -2 and Table -3 gives a comparison between RS encoder and decoder on Spartan 3E and Spartan 6 FPGA kits. Here it shows that Spartan 6 on which we have worked used less area as compared to Spartan 3E kit. This fulfills our goal of obtaining the optimized area.

Table -2: Comparison of RS encoder on Spartan 3E and Spartan 6

	RS Encoder		RS Encoder	
	Spartan 3E (XC3S1200E, package fg320,speed grade-5)		Spartan 6(XC6SLX16, package cgs324, speed grade-3)	
Logic Utilization	Used	Utilization	Used	Utilization
No. of slices	251	2%	149	0%
No. of slices f/fs	274	1%	142	66%
No. of 4 input LUTs	461	2%	206	0%
No. of bonded IOBs	21	8%	23	5%
No. of GCTRLs	1	4%	1	3%

Table -3: Comparison of RS decoder on Spartan 3E and Spartan 6

	RS Decoder		RS Decoder	
	Spartan 3E (XC3S1200E, package fg320,speed grade-5)		Spartan 6(XC6SLX16, package cgs324, speed grade-3)	
Logic Utilization	Used	Utilization	Used	Utilization
No. of slices	775	8%	592	2%
No. of slices f/fs	517	2%	368	50%
No. of 4 input LUTs	1459	8%	508	1%
No. of bonded IOBs	19	7%	21	4%
No. of GCTRLs	2	8%	1	3%

7. CONCLUSIONS

In this paper, error detection and correction techniques have been used which are essential for reliable communication over a noisy channel. The effect of errors occurring during transmission is reduced by adding redundancy to the data prior to transmission. The redundancy enables a decoder to detect and correct errors. Cyclic linear block codes are used efficiently for error detection and correction. The encoder splits the incoming data stream into blocks and processes each block individually by adding redundancy according to the prescribed algorithm. Likewise, the decoder individually processes each block and it corrects errors by using the redundancy present in the received data. An important advantage of cyclic codes is that they are easy to encode. Also they possess a well defined mathematical structure which has lead to very efficient decoding schemes for them. Hence at last the design written in VHDL is successfully implemented on Spartan 6 FPGA.

ACKNOWLEDGEMENT

The author would like to thank Shri Sant Gajanan Maharaj College of Engineering for providing this opportunity to do work in this field and also want to thank faculty members for continuous guidance and encouragement.

REFERENCES

[1] Abhinav Agarwal, Man Cheuk Ng, and Arvind, “A Comparative Evaluation of High-Level Hardware Synthesis Using Reed–Solomon Decoder”, IEEE Embedded

Systems Letters, VOL. 2, No. 3, September 2010.

[2] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, "Concurrent Error Detection in Reed–Solomon Encoders &Decoders", IEEE Transactions on very large scale integration (VLSI) systems, VOL. 15, No. 7, July 2007

[3] Rajeev Kumar Patial and Priyanka Dayal, "FPGA Implementation of Reed-Solomon Encoder and Decoder for Wireless Network 802.16", Int. Journal of Computer Appl's (0975–8887) Vol 68– No.16,April 13.

[4] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsan, "Analysis of Errors and Erasures in Parity Sharing RS Codecs", IEEE transactions on computer VOL. 56, No. 12, December 2007.

[5] Diplaxmi Chaudhari, Mayura Bhujade and Pranali Dhumal, "VHDL Design and FPGA Implementation of Reed Solomon Encoder and Decoder for RS (7, 3)", International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 3, March 2014 563.

[6] Aqib Al Azad and Md Imam Shahed, "A Compact and Fast FPGA Based Implementation of Encoding and Decoding Algorithm Using Reed Solomon Codes", International Journal of Future Computer and Communication, Vol. 3, No. 1, Feb. 2014.

[7] Sandeep Kaur, "VHDL implementation of Reed Solomon Codes", Thapar Institute of Engineering and Technology, Patiala, 2006.

[8] Hazem Abd Elall Ahmed Elsaid, "Design and Implementation of Reed Solomon Decoder using Decomposed Inversion less Berlekamp-Massey Algorithm", Faculty of Engineering, Cairo University Giza, Egypt,2010.

[9] Harikishore Kakarla, Madhavi Latha and Habibulla Khan, "Optimal Self Correcting Fault Free Error Coding Technique in Memory Operation", International Journal of Computer Science & Information Technology (IJCSIT), Vol.3, No.3,June 2011.

[10] Zi-Yi Lam, Wai-Leong Pang, Chee-Pun Ooi, Sew-Kin Wong and Kah-Yoong Chan, "VHDL Modelling of Reed Solomon Decoder", Research Journal of Applied Sciences, Engineering and Technology 4(23): 5193-5200, 2012 ISSN: 2040-7467© Maxwell Scientific Organization, 2012.

[11] S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields", SIAM Journal of Applied Maths, vol.8, pp.300–304

[12] R. J. McEliece, "Finite Fields for Computer Scientists and Engineers", Boston, MA: Kluwer Academic, 1987.

[13] S. B. Wicker, "Error Control Systems for Digital Communication and Storage", Englewood Cliffs, N.J.:Prentice-Hall, 1994.

[14] M. Kaur and V. Sharma, "Study of Forward Error Correction using Reed—Solomon Codes", International Journal of Electronics Engineering, vol. 2, pp. 331 – 333, 2010.

[15] M. Purser, "Introduction to Error Correcting Codes", Artech House, Boston-London, 1995.

[16] K. Sam. Shanmugam, "Digital and Analog Communication System", The University of Michigan, Wiley, 1979.

[17] C. K. P. Clarke, "Reed-Solomon Error Correction**",** BBC Research & Development, White Paper WHP 031.

Author Profile

Harshada L. Borkar , is born on 16th Dec 1989 in Nagpur, Maharashtra. She received the B.E. degree in Electronics and Telecommunication Engineering from Datta Meghe Institute of Engineering Technology and Research and M.E. degree in Digital Electronics Engineering from Sant Shri Gajanan Maharaj College of Engineering in 2012 and 2015, respectively.