A STUDY & VHDL IMPLEMENTATION OF REED SOLOMON ERROR CORRECTING CODES

[1]Sanjeev Kumar, [2]Ms. Priyanka

[1]M.tech Scholar, [2]Assistant Professor, Department of ECE
Applied College of Management and Engineering, Mitrol (Palwal),
Maharshi Dayanand University, Rohtak, Haryana (India)

ABSTRACT

In the present world, communication system which includes wireless, satellite and space communication, reducing error is being critical. During message transferring the data might get corrupted, so high bit error rate of the wireless communication system requires employing to various coding methods for transferring the data. Channel coding for detection and correction of error helps the communication systems design to reduce the noise effect during transmission [1]. In this paper, Reed Solomon (RS) Encoder and Decoder and their VHDL implementation using ModelSim tool is analyzed. RS codes are non- binary cyclic error correcting block codes. Here redundant symbols are generated in the encoder using a generator polynomial g(x) and added to the very end of the message symbols. Then RS Decoder determines the locations and magnitudes of errors in the received polynomial. The paper covers the RS encoding and decoding algorithm, simulation results.

***Keywords*: Reed-Solomon (RS), Galois Field (GF), Generator Polynomial g(x), Block length, VHDL (VHSIC Hardware description Language).**

I INTRODUCTION

REED Solomon (RS) codes [3], encoders and decoders are extremely powerful error correcting tools that increase transmission quality to a great extent. RS codes operate on the information by dividing the message stream into blocks of data, adding redundancy per block depending only on the current inputs. The symbols in RS coding are elements of a finite field or Galois Field (GF). A GF is a set that consists of finite number of elements. Encoding is achieved by affixing the remainder of a GF polynomial division into the message. This division is accomplished by a Linear Feedback Shift Register (LFSR) implementation. The mathematics of RS encoding is based on finite field arithmetic [4], [5]. GF multipliers are used for encoding the information block. The function of the decoder is to process the received codeword to compute an estimate of the original message symbols. Typically about ten times more resources are required to decode and correct the corrupted data. The codes are represented by the format RS (n, k) where n is the total number of s-bit wide symbols, and k is the number of s-bit wide information (data) symbols in a codeword. RS Decoder performs detection and correction of information (data) symbols in a codeword. The RS encoded data is processed to determine whether any errors have occurred during transmission. Once the number of errors is determined, the decoder decides if they are within the range of correction. After determining this, the decoder corrects the

errors in the received data.

RS codes are generally represented as an RS (n, k), with m-bit symbols, where

Block Length :	n
No. of Original Message symbols :	k
Number of Parity Digits :	n - k = 2t
Minimum Distance :	d = 2t + 1.

A codeword based on these parameters is shown diagrammatically in Figure

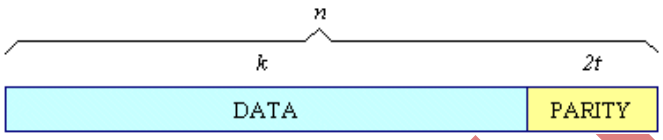


Fig. 1 The Structure of a RS Codeword [6]

Such a code can correct up to (n-k)/2 or t symbol (each symbol is an element e of GF ($2^{(m)}$) (i.e.) any t symbols corrupted in anyway (single- or multiple-bit errors) can still lead to recovery of the original message. RS codes are one of the most powerful burst - error correcting codes and have the highest code rate of all binary codes. They are particularly good at dealing with burst 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.

A Reed Solomon protected communication or data transfer channel is as shown in Figure 2

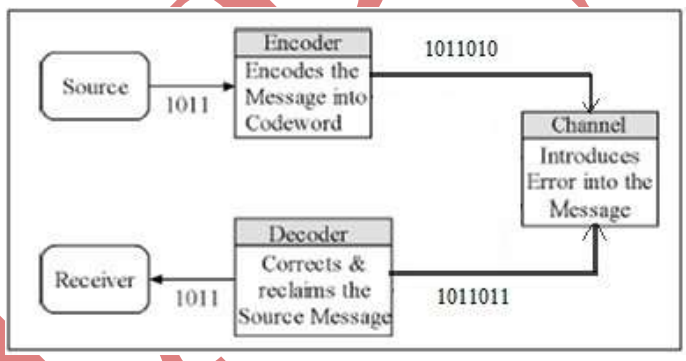


Figure 2: Reed Solomon Protected Channel [2]

This paper presents an efficient design and VHDL implementation of RS Encoder and Decoder. The following Chronology is being followed to present the paper. In section II, the operation and architecture of RS Encoder is discussed. Then in section III, RS Decoding algorithm is given. Section IV deals with the implementation details of basic blocks for both RS Encoder and Decoder. Section V deals with simulation results. Section VI deals with Properties of RS codes and sec. VII has the conclusion.

II RS ENCODER

The Reed Solomon Encoder reads in k data symbols computes the n – k symbols, append 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 coefficient produced will be symbols such that the generator polynomial will

exactly divide the data/parity polynomial. [7]

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.

$$c(x) = m(x) * 2t + m(x) \bmod g(x) \quad (1)$$

Where, g(x) is the generator polynomial of degree 2t and given by,

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

2.1 Operation

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 [7], [8], [9].

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. Encoder architecture is shown in Fig. 2.

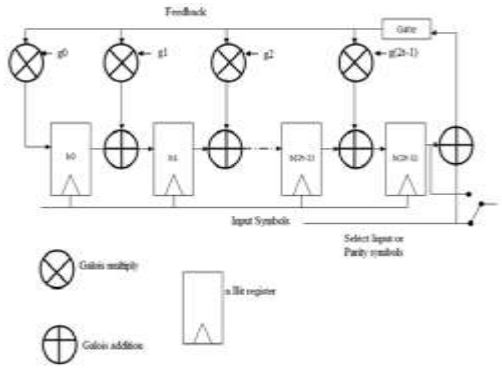
The Encoder architecture shows that one input to each multiplier is a constant field element, which is a coefficient of the polynomial g(x). For a particular block, the information polynomial M(x) is given into the encoder symbol by symbol. These symbols appear at the output of the encoder after a desired 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 disable 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 [8].

2.2 Architecture

The encoder is architected using the Linear Feedback Shift Register Design. The coefficients g_i, $0 \leq i \leq 15$ are derived as,

$$\begin{aligned} g(x) = {} & x^{16} + 59x^{15} + 13x^{14} + 104x^{13} + 189x^{12} + 68x^{11} \\ & + 209x^{10} + 30x^{9} + 8x^{8} + 163x^{7} + 65x^{6} \\ & + 41x^{5} + 229x^{4} + 98x^{3} + 50x^{2} + 36x + 59 \end{aligned} \quad (3)$$

Each message is accompanied by a pulse signal, which indicates the beginning of a message. After 239 clock cycles, the encoder starts concatenating the 16 calculated parities to the message to make a codeword of 255 symbols [10].



.

Fig. 3 Architecture of RS Encoder

III RS DECODER

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 (x^2, x^3, 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 [11].

3.1 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 [12]. 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.

The first step is to calculate the syndrom values [13] from the received codeword. These are then used to find the coefficients of the error locator polynomial Λ_1 ....Λ_v and the error magnitude polynomial Ω_0 .... Ω_{v-1} using the Euclidean algorithm [14]. The error locations are figured out by the Chien search [15] and the error

magnitudes are calculated using Forney's method [16]. The error magnitude vector Y comes out of the Chien/Forney block in reverse order, so it is passed through a FIFO block before it is added to the received codeword R(x). As these calculations involve all the symbols of the received code word, it is necessary to restore the message until the results of the calculation are available. Then, to correct the errors, each error value is added (modulo 2) to the symbol at the appropriate location in the received codeword [17].

3.2 Architecture

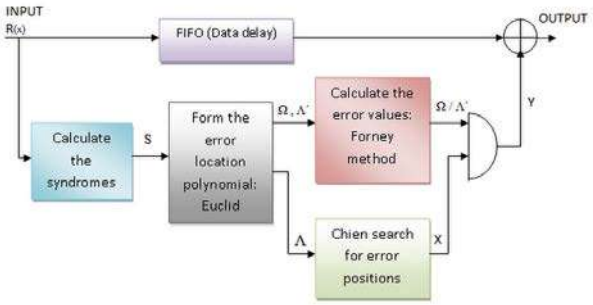


Fig. 4 General Architecture of RS Decoder [13]

IV. IMPLEMENTATION DETAILS OF BASIC BLOCKS

4.1 Design hierarchy

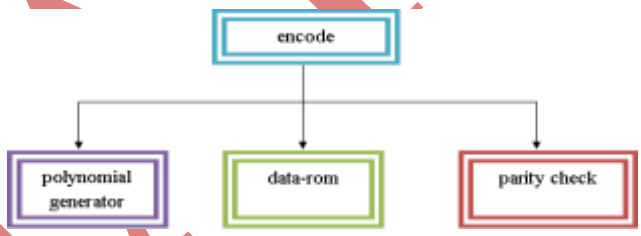


Fig. 5 Design hierarchy for RS Encoder

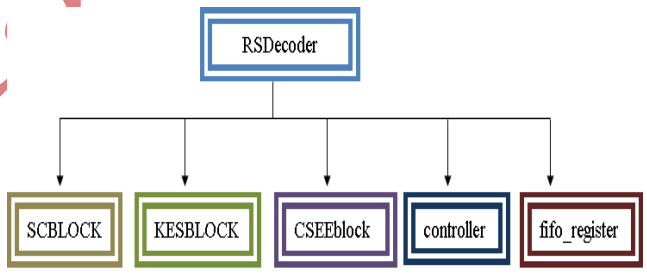


Fig. 6 Design hierarchy for RS Decoder

4.2 Block description of RS Encoder

1. Polynomial generator: For the design we used the following field generator polynomial

$$p(x) = x^8 + x^4 + x^3 + x^2 + x^1 + 1 \quad (4)$$ and the generator polynomial

$$g(x) = g_0 + g_1(x) + g_2(x) + \cdots + g_{2t-1}(x) + x \quad (5)$$

2. Data-rom: We provide a ROM as the message to encode.

3. *Parity check:* The parity check block allows the encoder to output parity symbols.

4.3 Block description of RS Decoder

1. SCBlock: At the end of received word, all cells store the syndrome values and Syndrome Computation (SC) block sets its flag high if one or more syndrome values are not zero.

2. KESBlock: The Key equation solver (KES) block provides two polynomial -error locator polynomial and error magnitude polynomial.

3. CSEEBlock: Chien Search and Error Evaluator (CSEE) block identifies error location while computes its error magnitude.

4. FIFO register: FIFO register stores 32 received word symbols. To match the order of the bytes in error vector and received codeword FIFO is applied as the error vector is produced in the reverse order of the received codeword.

5. Controller: The controller provides synchronization among all four modules -SC, KES, CSEE and FIFO Registers.

V RESULTS

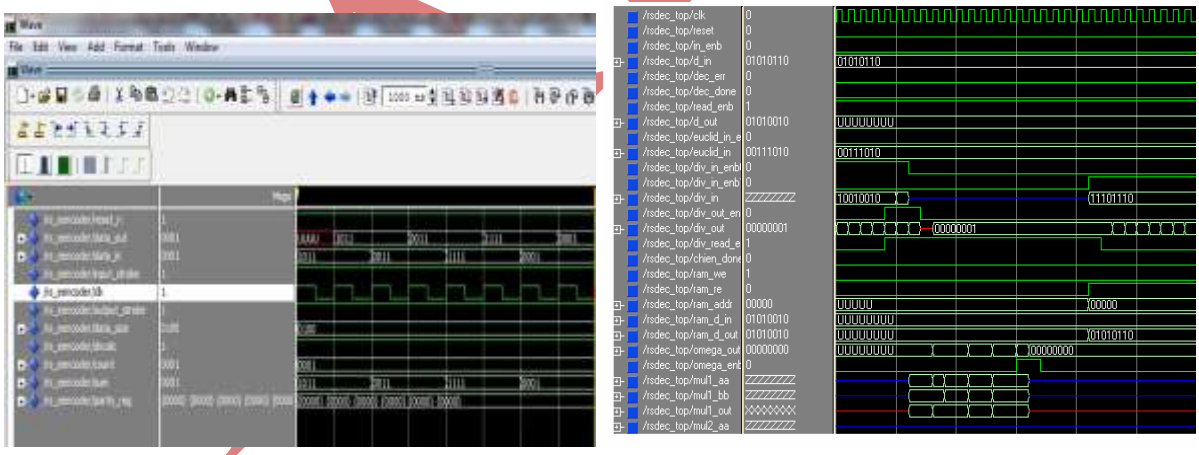


Fig. 7 and Fig. 8 describe the simulation waveform of RS Encoder and Decoder.

Using interleaving, the burst error channel is effectively transformed into an independent error channel for which many FEC coding techniques are applicable. An interleaver scrambles the encoded data stream in a deterministic manner by separating the successive bits transmitted over the channel as widely as possible. In the deinterleaver, the inverse operation is preformed and the received data is unscrambled so that the decoding operation may proceed properly. After deinterleaving, error bursts that occur on the channel are spread out in the data sequence to be decoded, thereby spanning many code words.

The combination of interleaving and FEC thus provides an effective means of combating the effect of error bursts.

VI PROPERTIES OF REED SOLOMON CODES

- It does not matter to the code how many bits in a symbol are incorrect, 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. [1]
- Designers are not required to use the natural sizes of Reed-Solomon code blocks. A technique known as "shortening" produces 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). At the decoder, the same portion of the block is loaded locally with binary zeroes [6].
- A Reed–Solomon code operating on 8-bits symbols has n = $2^8 - 1 = 255$ symbols per block because the number of symbol in the encoded block is $n = 2m - 1$.
- For the designer its capability to correct both burst errors makes it the best choice to use as the encoding and decoding tool. [2]

VII CONCLUSIONS

In this work an efficient VHDL implementation of RS Encoder and Decoder was presented. Here error detection and correction techniques have been used which are essential for reliable communication over a noisy channel. The results demonstrate that the Reed Solomon codes are very efficient for the detection and correction of burst errors. 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. This design will play a remarkable role with its significant speed and efficiency.

REFERENCES

[1] Priyanka Srivastava, *"Error Detection and Correction Using Reed Solomon Codes", IJARSCCE* Volume 3, Issue 8, August 2013.

[2] Yo Sup (Joseph) Moon, "*Introduction to Reed-Solomon Codes"* , Harvard University Department of Mathematics, August 2011.

[3] I.S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *SIAM Journal of Applied Mathematics*, vol. 8, 1960, pp. 300– 304.

[4] R. J. McEliece, *Finite Fields for Computer Scientists and Engi*neers. Boston, MA: Kluwer Academic, 1987.

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

[6] 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.

[7] "Reed-Solomon (RS) Coding Overview," VOCAL Technologies, Ltd., Rev. 2.28n, 2010.

[8] S. Kaur "VHDL Implementation of Reed-Solomon code," Thesis, Thapar Institute of Engg, 2006.

[9] J. Y. Chang and C. B. Shung, " A high speed Reed-Solomon codec chip using look forward architecture," *IEEE APC CAS'94*, pp. 212-217, Dec. 1994.

[10] *K. C. C.* Wai and S. J. Yang, "*Field Programmable Gate Array Implementation of Reed-Solomon Code, RS (255, 239)*", New York (2006).

[11] S. S. Shah, S. Yaqub, and F. Suleman, "Self-correcting Codes Conquer Noise Part 2: Reed-Solomon Codecs," *EDN*, Mar. 15, 2001, pp. 107- 120.

[12] B. Sklar. *Digital Communications: Fundamentals and Applications*, 2nd ed., Upper Saddle River, NJ: Prentice-Hall, 2001. p. 441.

[13] C.K.P. Clarke. Reed-Solomon error correction, BBC R&D White Paper,WHP 031, July 2002.

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

[15] R. T. Chien. "Cyclic Decoding Procedure for the Bose- Chaudhuri- Hocquenghem Codes." *IEEE Transactions on Information Theory*, Volume IT-10, pp. 357-363, Oct. 1964.

[16] G. D. Forney. "On Decoding BCH Codes." *IEEE Transactions on Information Theory*, Volume IT-11, pp. 549-557, Oct. 1965.

[17] E. R. Berlekamp, R. E. Peile and S. P. Pope. "The Application of Error Control to Communications." *IEEE Communications Magazine*, vol. 25, no. 4, pp. 44-57, Apr. 1987.