Design and Implementation of Reed Solomon Encoder on FPGA

Amandeep Singh, Mandeep Kaur

Abstract—Error correcting codes are used for detection and correction of errors in digital communication system. Error correcting coding is based on appending of redundancy to the information message according to a prescribed algorithm. Reed Solomon codes are part of channel coding and withstand the effect of noise, interference and fading. Galois field arithmetic is used for encoding and decoding reed Solomon codes. Galois field multipliers and linear feedback shift registers are used for encoding the information data block. The design of Reed Solomon encoder is complex because of use of LFSR and Galois field arithmetic. The purpose of this paper is to design and implement Reed Solomon (255, 239) encoder with optimized and lesser number of Galois Field multipliers. Symmetric generator polynomial is used to reduce the number of GF multipliers. To increase the capability toward error correction, convolution interleaving will be used with RS encoder. The Design will be implemented on Xilinx FPGA Spartan II.

Keywords—Galois Field, Generator polynomial, LFSR, Reed Solomon.

I. Introduction

DIGITAL Communication system is immune to errors, but there exit some errors which result in wrong data reception by receiver. Thus, transmitter has to send same data again to compensate for errors. This results in wastage of resources. To reduce the burden over transmitter to transmit same data again error correcting codes are used for detection and correction of errors that are introduced during transmission of data from source to destination. Reed Solomon codes are non-binary BCH error correcting codes.In 1959, Irving Reed and Gus Solomon described a new class of error-correcting codes called Reed-Solomon codes [1].Originally Reed-Solomon codes were constructed and decoded through the use of finite field arithmetic. Reed Solomon codes are burst error correcting codes. RS codes detect and correct errors on symbol level i.e. if there is any error of 1-bit or 2-bit or m bit in symbol of m bit then these error correcting codes will correct the complete symbol. Before the transmission of data, RS encoder appends some parity bits to the data so that at decoder these bits can be used for error detection and correction.

Algorithm used for encoding RS codes is very complex as calculations are done over Galois field. The complexity can be reduced with use of symmetric coefficients of generator polynomial [2]. Further, multiplication values can be optimized which reduces the number of AND gates and multiplication can be implemented with XOR gates only if one of the operand is known [3].

Objective of this work is to implement Reed Solomon encoder with symmetric generator polynomial and globally optimized Galois Field multipliers [4]. The design was implemented on Xilinx Spartan 2e FPGA.

This paper goes over with theory behind Reed-Solomon encoding, Architecture of implemented RS encoder and future scope of the study.

II. Basics Of Reed Solomon FEC Codes.

Reed Solomon codes are forward error correcting codes that can be specified as RS (n,k) where n is the size of code word generated by RS encoder and k is size of data input to RS encoder. The difference between number of symbols out from RS encoder and number of symbols input to RS encoder is called parity symbols 2t. Each symbol is formed from m bits. Fig 1 below shows the code word of Reed Solomon code.

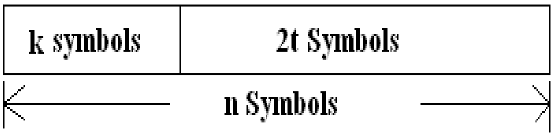

Fig. 1 Code word generated by Reed Solomon encoder
k: Symbols input to RS encoder; n: Symbols output from RS encoder ($n = 2^m - 1$); 2t: parity symbols = n – k; m: Size of each symbol

Data stream of bits is divided into k symbols each being m bit long. 2t parity symbols are appended to k symbols to give n symbols of RS code. Data symbol is represented in polynomial form with highest power of x representing MSB and lowest power of x representing LSB. Input data symbols will contain powers from 0 to k – 1 (LSB to MSB) and output symbols will contain powers from 0 to n-1 (LSB to MSB) [5]. Reed Solomon codes are constructed using a special type of polynomial called generator polynomial*g(x)* having α as its one of root is given by (1) and (2).

$$g(x) = (x+\alpha)\,(x+\alpha^2)\,(x+\alpha^3)\,....\,(x+\alpha^{2t}) \qquad (1)$$

$$g(x) = \prod_{i=1}^{2t}(x + \alpha^{i}) \qquad (2)$$

The code word polynomial c(x) is generated by equation below:

Amandeep Singh is Postgraduate Student at Electronics and Communication Engineering department, UCoE, Punjabi University Patiala, Punjab, India (e-mail: amandeep.singh568@gmail.com)

Mandeep Kaur is Assistant Prof. at Electronics and Communication Engineering department, UCoE, Punjabi University Patiala, Punjab, India (e-mail: ermandeep0@gmail.com).

$$c(x) = g(x) \bullet d(x) \tag{3}$$

where d(x) is input data symbol polynomial.

RS codes generated by (3) are non-systematic. Systematic codes are those containing data bits along with parity bits. To generate a systematic Reed Solomon codes following procedure is followed [6]:

1. Multiply the information symbols with X^{n-k}. This can be done by shifting the information symbols to the left to allow space for 2t parity symbols.
2. Divide the result of step 1 with the generator polynomial using GF algebra.
3. Add the result of step 2 (remainder of division) to the result of step 1.

Thus a systematic RS code is given by (4):

$$C(x) = x^{n-k}\, d(x) + R(x) \tag{4}$$

$$where, R(x) = rem\left(\frac{x^{n-k}\, d(x)}{g(x)}\right) \tag{5}$$

Here, P(x) represents the parity symbols or check symbols. These parity symbols are appended with the data symbols to form code word C(x). Code word will be completely divisible by generator polynomial g(x).

III. Reed Solomon Encoder Architecture

Each RS code word will consist of 239 message symbols (m_0, m_1, … m_{238}), each symbol being of 8 bit long and 16 redundant parity symbols ($r_{0,}$ $r_{1,}$ … r_{15}). These parity symbols are generated as given by equation above. The proposed RS encoder architecture uses 8-bit symbols over Galois field GF (2^8) and primitive polynomial P(x) is

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

To reduce the number of multipliers needed, symmetrical generator polynomial proposed by Berlekamp [2] is used and is given by (6).

$$g(x) = \prod_{i=120}^{135}(x + \alpha^i) \tag{6}$$

where α is primitive element and root of primitive polynomial P(x).

A Reed Solomon consists of linear feedback shift registers labeled form b_0 to b_{2t-1}, Galois field multipliers and adders. LFSR helps in shifting and division operation. The complexity of RS encoder depends mainly on GF multipliers and adders. The number of multipliers is reduced using symmetric generator polynomial [2]. Further, the multiplication can be optimized by using method proposed in [3]. As one of the operand of the multiplier is constant so the multiplication operation can be implemented by using XOR gates only.

Let variable field element by A = (a_0, a_1, … a_7) and other element will be coefficient of generator polynomial which is constant. Thus, multiplication can be given by:

$$\alpha^{240}\begin{bmatrix} a_7 + a_6 + a_5 + a_1 \\ a_7 + a_6 + a_2 \\ a_7 + a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_0 \end{bmatrix} = a_7 + a_3 + a_5 + a_6$$

Thus if one of the operand is known then multiplication can be implemented with fever number of components. Thus only operation required for multiplication is XOR. Above example requires three XOR gates. But still there are some redundancies removing which can further reduce the number of XOR gates.

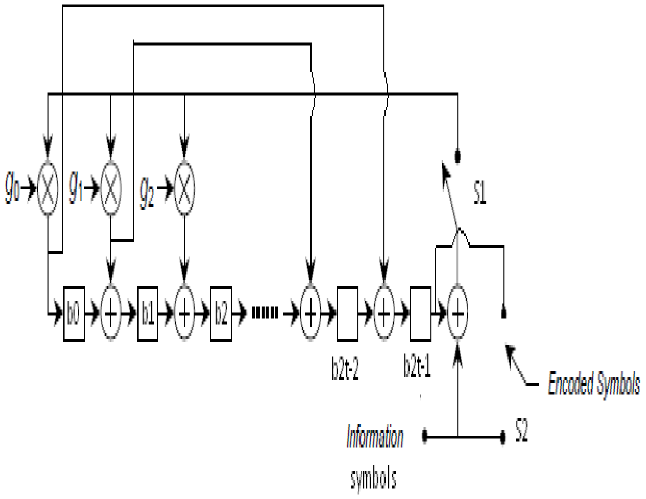

Fig. 2 Reed Solomon encoder with symmetric generator polynomial

To increase the reliability for error detection and correction, the RS output is convolution interleaved. The final output will be convolutional interleaved.

IV. Experimental Setup And Results

The above proposed design is implemented with help of Xilinx ISE 8.1i tool on Xilinx Spartan II FPGA device xc2s50pq208. The specifications of the RS encoder are shown in Table I.

TABLE I
Specifications of Design of RS Encoder

Number of information symbols	239
Number of Code word symbols	255
Number of parity symbols	16
Primitive polynomial	$x^8 + x^4 + x^3 + x^2 + 1$
Generator polynomial	$x^{16} + \alpha^{240} x^{15} + \alpha^{89} x^{14} + \alpha^{41} x^{13} + \alpha^{79} x^{12} + \alpha^7 x^{11} + \alpha^{33} x^{10} + \alpha^{151} x^9 + \alpha^{136} x^8 + \alpha^{151} x^7 + \alpha^{33} x^6 + \alpha^7 x^5 + \alpha^{79} x^4 + \alpha^{41} x^3 + \alpha^{89} x^2 + \alpha^{240} x + 1$

To verify the design, input to the design was given from PC serially with help of RS-232 protocol. Fig shows the experimental setup of the design.

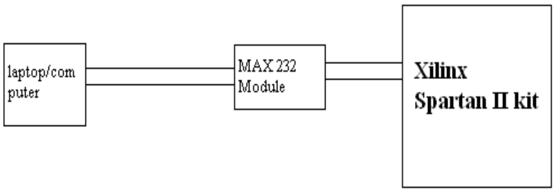

Fig. 3 Experimental setup

Max-232 module is a level converter which converts the RS232 levels to TTL. Input to design was given from HyperTerminal with baud rate of 9600bps. Fig. 4 shows the behavioral simulation of the design.

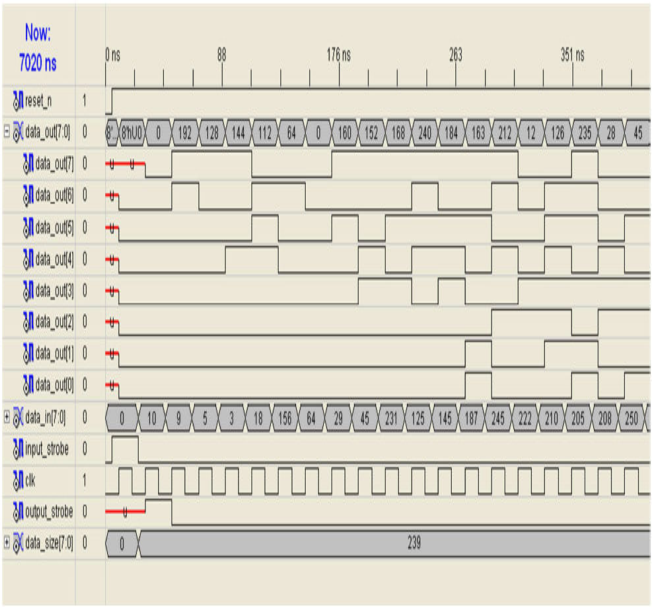

Fig. 4 Behavioral Simulation of RS encoder

Test bench designed showed and verifies the functionality of the design. As the design is optimized to lesser number of components so table shows that the proposed design uses lesser number of components when optimized as co, thus we can do a little compromise with power to gain better BERs [8]. Table II shows the device utilization of Xilinx Spartan II device.

TABLE II
DEVICE UTILIZATION OF RS ENCODER

Logic Utilization	Used	Available	Utilization
Slice Flip-flops	164	1,536	10%
4 input LUTs	249	1,536	16%
Logic distribution			
occupied Slices	145	768	18%
Total Number 4 input LUTs	261	1,536	16%
bonded IOBs	27	140	19%
GCLKs	1	4	25%
GCLKIOBs	1	4	25%
Total equivalent gate count for design	**4,422**		

V. CONCLUSION

In this paper 8 error correcting code RS(255, 239) encoder with optimized multipliers is presented. The proposed design uses symmetric coefficients of generator polynomial and multipliers are optimized to use XOR gates only thus giving a significant reduction in device utilization as compared to [7]. The element in FPGA which describes the complexity of design is LUTs, which are reduced in this design considerably. This results in reduction in cost, area and power consumption. The design is implemented in VHDL and synthesized on Xilinx Spartan II.

VI. REFERENCES

[1] I.S. Reed and G. Solomon, “polynomial Codes over Certain Finite Fields”, SIAM Journal of Applied Mathematics, Volume 8, 1960, pp.300-304.

[2] E. R. Berlekamp, "Better Reed-Solomon encoders," presented at California Inst. of Technol., Elec. Eng. Seminar, Pasadena, CA, Dec. 12. 1979.

[3] GeQun, Mao Junfa, and RongMengtian, “A VLSI Implementation of A Low Complexity Reed-Solomon Encoder and Decoder For CDPD”, ASIC 2001. Proceedings 4th international conference, 2001, pp. 435-439.

[4] J. Jittawutipoka and J. Ngarmnil, “Low complexity reed solomon encoder using Globally optimized finite field multipliers”, IEEE region 10 conference, vol. 4, Nov. 2004, pp. 423-426.

[5] Amandeep Singh and Mandeepkaur, “Study of Reed Solomon Encoder”, International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 2, April 2013

[6] Petrus Mursanto, “Generic Reed Solomon encoder”, Makara, Sains, Vol. 10, No. 2, Nov. 2006, pp. 58-62.

[7] Priyanka Dayal and Rajeev Kumar Patial, “FPGA Implementation of Reed-Solomon Encoder and Decoder for Wireless Network 802.16”, International Journal of Computer Applications, Volume 68– No.16, April 2013,pp. 42-45

[8] Lionel Biard, Dominique Noguet, “Reed-Solomon Codes for Low Power Communications”, Journal of Communications, Vol. 3, No. 2, April 2008, pp. 13-21.