Implementation of Reed-Solomon RS(255,239) Code

Maja Malenko
SS. Cyril and Methodius University - Faculty of Electrical Engineering and Information Technologies
Karpos II bb, PO Box 574, 1000 Skopje, Macedonia
E-mail: majam@feit.ukim.edu.mk

***Abstract*—In this paper the construction of Reed-Solomon RS(255,239) codeword is described and the process of coding and decoding a message is simulated and verified. RS(255,239), or its shortened version RS(224,208) is used as a coding technique in Low-Power Single Carrier (LPSC) physical layer, as described in IEEE 802.11ad standard. The encoder takes 239 8-bit information symbols, adds 16 parity symbols and constructs 255-byte codeword to be transmitted through wireless communication channel. RS(255,239) codeword is defined over Galois Field GF(2^8) and is used for correcting up to 8 symbol errors. RS(255,239) code construction is fully implemented and Simulink test project is constructed for testing and analyzing purposes.**

***Keywords:* galois field, IEEE 802.11ad, QAM, Reed-Solomon**

I. Introduction

Reed-Solomon codes are powerful technique for error detection and correction. Reed-Solomon codes are representatives of block code family, as they operate on an information by dividing it into blocks of data, consisting of information symbols to which parity symbols are added [1]. Reed-Solomon codes are known as RS(n,k) codes, where n represents the total number of symbols in one code block, while k is the number of information symbols in that block.

IEEE 802.11ad standard is popular standard for wireless local area transmission at 60 GHz with very high throughput. The IEEE 802.11ad standard supports three different modulation methods: spread-spectrum modulation, single carrier (SC) modulation and orthogonal frequency division multiplex (OFDM) modulation [2]. Different coding schemes are defined for each of these modulation methods, each having different error protection coding complexity and performance. One of the modulation methods which are covered by this standard is the Low-Power Single Carrier modulation technique, as subset of Single Carrier modulation.

The purpose of this paper is to fully describe and implement the coding scheme used in Low-Power Single Carrier, which is RS(255,239) coding. It will be tested and simulated in Simulink.

The block diagram of Low-Power Single Carrier encoding and modulation steps is given in Fig. 1. Our focus will be RS (255,239) encoder block, which is simplified alternative version of LDPC encoding used in Single Carrier. Low-Power Single Carrier uses this encoding mainly because of the minimized power consumption, but still at the expense of less robust error correction.

The paper is being organized in the following chronology. First the Reed-Solomon encoding technique is being discussed, along with the necessary mathematical background needed for construction of RS(255,239) codes. In section III the process of encoding an example 239-byte message is described. In sections IV and V the encoding process is being simulated and verified in Simulink. Lastly, a conclusion is being drawn.

II. Reed-Solomon Encoding Technique

Reed-Solomon codes are used as an error correction technique which adds redundant information to the data being transferred. The receiver, again using Reed Solomon decoding algorithm, can detect and correct specified number of errors which may have occurred in transmission. Reed-Solomon codes are widely used error correcting codes, mainly because of their efficiency and simplicity, compared to other codes [3].

First, some essential background on Reed-Solomon coding theory is provided, along with Galois field arithmetic which is the base for generating Reed-Solomon codes. Then, the process of encoding SC information, with given parameters is described, and simulated using Simulink.

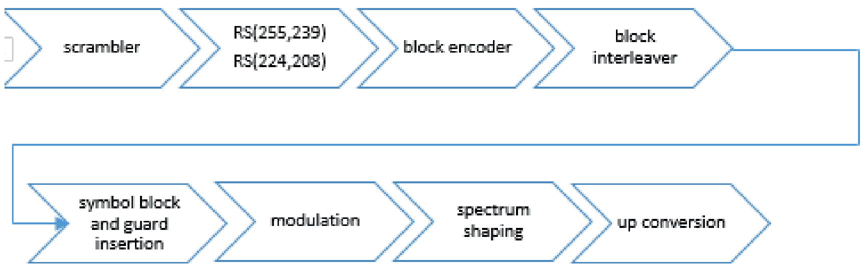

Fig. 1. Block diagram of Low-Power Single Carrier encoding and modulation steps.

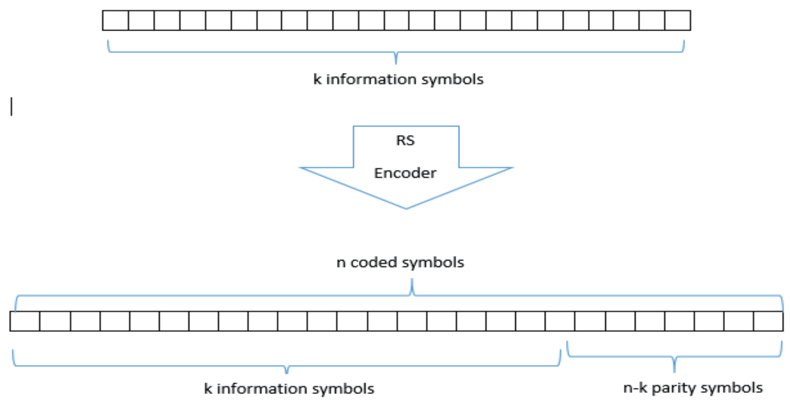

Fig. 2. Reed-Solomon code.

A. Reed-Solomon Codes

Every error detection and correction code has three primary characteristics: code length, dimension and minimum distance. The code length, n, represents the number of symbols per codeword. The dimension of the code, k, represents the number of actual information symbols, transmitted in one codeword. The minimum distance is the minimum number of symbol differences between codewords.

Reed-Solomon codes are block codes, which means they operate on blocks of data. They have linear and cyclic characteristics. Linearity is ability to construct new codeword by adding together already constructed codeword, while the cyclic characteristic is seen as the ability to produce new codeword by cyclically shifting the symbols of a given codeword. Reed-Solomon codes are particularly well suited for detecting and correcting burst data errors. If a symbol has error in more than one bit, that error still counts as one symbol error that can be corrected. This leads to a conclusion that Reed-Solomon codes can correct many bit errors.

The process of encoding a message consists of dividing the message which needs to be transmitted into submessages of specified length. Then, parity protection information is added to the end of each submessage, thus forming one block of specified length (specified number of symbols).

Reed-Solomon codes use abbreviation RS(n,k), where n is the number of symbols in one block of data, and k is the number of information symbols in that block. If the number of bits in one symbol is m, then the number of symbols in a block can't be more than $2^m - 1$. In each block there are n-k parity symbols, which means that the maximum number of corrected errors(in terms of symbols) in a block can be t = (n-k)/2, as shown in Fig 2.

Different values of parameters n and k, provide different error correcting capabilities and produce different complexity of the whole error correcting system. In this paper we are focusing on RS(255,239) error correcting code, which can correct up to 8 erroneous symbols in one codeword. In the worst case, 8 bit errors may occur in one codeword, each one in different symbol, so the decoder can detect and correct 16 bit errors. In the best scenario, 8 complete symbol errors may occur, in every bit of 8 symbols, so the decoder can correct 8 x 8 bit errors.

Reed-Solomon codes perform their function according to finite field arithmetic, as presented in the next section.

B. Galois Fields

Galois Field is a finite field, consisting of a finite number of primitive elements, usually denoted as α. If a symbol length is m, then the number of elements in a Galois field will be 2^m. The field is abbreviated as GF(2^m) [4].

Each GF(2^m) element can be represented in two m-bit forms, as α coefficient which is an element from the set in (1) or as a polynomial expression, as described as in (2).

$$0, \alpha^0, \alpha^1, \dots, \alpha^{2^m-2} \tag{1}$$

$$a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 \tag{2}$$

The coefficients $a_{m-1} \dots a_1\ a_0$ can take only binary values.

One symbol in RS(255,239) consists of 8 bits, which means that Galois Field has 2^8, or 256 field elements.

In GF(255,239) there are 256 α coefficients:

$$0, \alpha^0, \alpha^1, \dots, \alpha^{254}$$

and the polynomial representation of a field element in the same Galois Field is:

$$a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

In order to define Reed-Solomon code, two main generator polynomials need to be constructed: field generator polynomial and code generator polynomial [5].

The field generator polynomial, p(x), needs to be irreducible polynomial of degree m. In GF(2^8) there are 18 irreducible polynomials, and we have chosen the primitive polynomial

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{3}$$

In order to define the Galois Field, we need to calculate all field elements. Given the primitive polynomial, p(x), and knowing that α must be a root of p(x), we can write:

$$p(\alpha) = 0$$
$$\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1 = 0 \tag{4}$$

$$\alpha^8 = \alpha^4 + \alpha^3 + \alpha^2 + 1$$

This means that α^8, which is one of 256 α coefficients, represents a Galois Field element written in binary form as 00011101 or in decimal form as 29.

By using Galois arithmetic operations, we can calculate the other 255 Galois Field elements as shown in Table 1.

TABLE I

GF(2^8) FIELD ELEMENTS, WITH $p(x) = x^8 + x^4 + x^3 + x^2 + 1$

Index form	Polinomial and binary form								Decimal form
	α^7	α^6	α^5	α^4	α^3	α^2	α^1	α^0	
0	0	0	0	0	0	0	0	0	0
α^0	0	0	0	0	0	0	0	1	1
α^1	0	0	0	0	0	0	1	0	2
α^2	0	0	0	0	0	1	0	0	4
α^3	0	0	0	0	1	0	0	0	8
α^4	0	0	0	1	0	0	0	0	16
α^5	0	0	1	0	0	0	0	0	32
α^6	0	1	0	0	0	0	0	0	64
α^7	1	0	0	0	0	0	0	0	128
α^8	0	0	0	1	1	1	0	1	29
α^9	0	0	1	1	1	0	1	0	58
α^{10}	0	1	1	1	0	1	0	0	116
α^{11}	1	1	1	0	1	0	0	0	232
. . .									
α^{253}	0	1	0	0	0	1	1	1	71
α^{254}	1	0	0	0	1	1	1	0	142

C. *Constructing RS(255,239)*

In order to represent the information and parity symbols of

Reed-Solomon code as elements of GF(2^8), the code generator polynomial, g(x), needs to be constructed. The code generator polynomial has 2t=n-k factors, and their roots are consecutive elements of previously defined Galois Field. In general form, the code generator polynomial is defined as:

$$g(x) = (x + \alpha^b)(x + \alpha^{b+1}) \dots (x + \alpha^{b+2t-1}) \qquad (5)$$

where b can be 0 or 1.

For RS(255,239) code, the number of parity symbols is n-k=2t=16, so the polynomial and decimal form of the code generator polynomial for correcting up to 8 errors is:

$$g(x) = (x + \alpha^0)(x + \alpha^1)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4)(x + \alpha^5)(x + \alpha^6)(x + \alpha^7)(x + \alpha^8)(x + \alpha^9)(x + \alpha^{10})(x + \alpha^{11})(x + \alpha^{12})(x + \alpha^{13})(x + \alpha^{14})(x + \alpha^{15}) = (x + 1)(x + 2)(x + 4)(x + 8)(x + 16)(x + 32)(x + 64)(x + 128)(x + 29)(x + 58)(x + 116)(x + 232)(x + 205)(x + 135)(x + 19)(x + 38) \qquad (6)$$

Galois Field multiplication and addition operations [6] are performed over g(x), and the resulting field generator polynomial, represented in decimal form, is:

$$g(x) = x^{16} + 118x^{15} + 52x^{14} + 103x^{13} + 31x^{12} + 104x^{11} + 126x^{10} + 187x^9 + 232x^8 + 17x^7 + 56x^6 + 183x^5 + 49x^4 + 100x^3 + 81x^2 + 44x^1 + 79 \qquad (7)$$

III. REED-SOLOMON ENCODING PROCESS

In this section the arithmetic Reed-Solomon encoding process will be briefly described, and then applied over an example sending message encoded using RS(255,239) code.

A. *Forming The Codeword*

The message that needs to be encoded in one block, consists of k information symbols. It can be represented as an information polynomial, I(x), of degree k-1:

$$I(x) = I_{k-1}x^{k-1} + \dots + I_1 x + I_0 \qquad (8)$$

where each of the coefficients $I_{k-1}, \dots, I_1, I_0$ is an m-bit symbol in GF(2^m).

The process of encoding the message represented in (8) consists of multiplying the information polynomial, I(x), by x^{n-k}, and then dividing the result by the code generator polynomial, g(x).

The encoded message polynomial, which represents the transmitted codeword, can be formed as follows:

$$T(x) = I(x)x^{n-k} + r(x) \qquad (9)$$

where r(x) is the remainder of division operation of $I(x)x^{n-k}$ with g(x).

B. *Encoding An Example RS(255,239) Message*

In order to produce RS(255,239) codeword, an information message consisting of 239 8-bit symbols was generated in Matlab. The information polynomial is in the following form:

$$x^{238} + 2x^{237} + 3x^{236} + \dots + 237x^2 + 238x + 239 \qquad (10)$$

This information polynomial is then multiplied by x^{16} to allow space for 16 parity symbols, and the following polynomial is produced:

$$x^{254} + 2x^{253} + 3x^{251} + \dots + 237x^{14} + 238x^{15} + 239x^{16} \qquad (11)$$

This polynomial is divided by the code generator polynomial, given in (6). The remainder represents the 16 parity symbols which are concatenated at the end of the information message, thus producing RS(255,239) codeword.

The encoded message polynomial, T(x), has the following form:

$$T(x) = x^{254} + 2x^{253} + 3x^{252} + \dots + 237x^{18} + 238x^{17} + 239x^{16} + 37x^{15} + 133x^{14} + 255x^{13} + 126x^{12} + 37x^{11} + 59x^{10} + 132x^9 + 133x^8 + 56x^7 + 168x^6 + 179x^5 + 4x^4 + 9x^3 + 99x^2 + 79x + 148 \qquad (12)$$

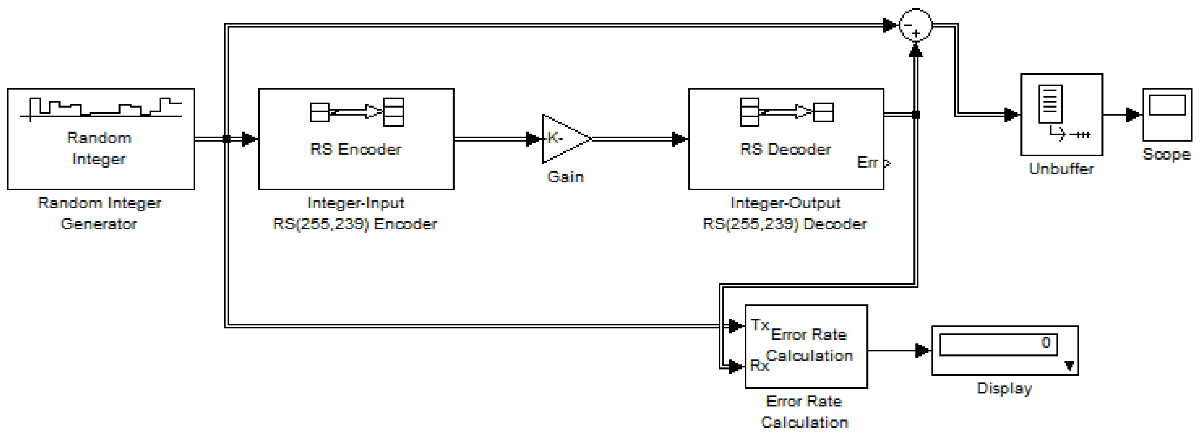

Fig. 3. RS(255,239) elementary system.

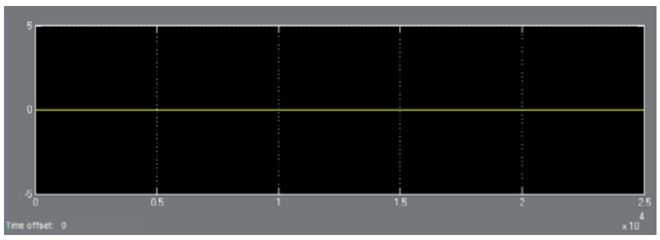

Fig. 4. Scope block plot.

or expressed as GF(2^8) coefficients:

1, 2, 3, …, 237, 238, 239, 37, 133, 255, 126, 37, 59, 132, 133, 56, 168, 179, 4, 9, 99, 79, 148.

IV. Testing The RS(255,239) Code

RS(255,239) encoding and decoding process was designed and tested in Simulink. Fig. 3 shows the elementary building blocks which are configured in order to perform RS(255,239) coding, without adding noise in the transmitted information. This model will be upgraded in the next section.

Random Integer Generator block produces information symbols which need to be encoded. Mary number value is set to 255 and samples per frame are to 239. The sample time is 1 second for every information symbol. Integer-Input RS Encoder encodes the information symbols, by adding 16 parity symbols. The codeword length value, N, is set to 255, and the message length, K, is set to 239. The primitive polynomial is specified as [1 0 0 0 1 1 1 0 1], which is binary representation of $p(x) = x^8 + x^4 + x^3 + x^2+1$, while the generator polynomial is calculated with specified Matlab function rsgenpoly(255,239). Gain block is set to ones for the first 239 coefficients and the other 16 parity symbols are set to zeros.

Integer-Output RS Decode block has identical settings as the encoder. The Scope block shows the difference between the original message and the recovered message. As can be seen from Fig. 4, the plot is zero, which confirms that the decoder successfully corrected the errors. The calculated error rate between the information data and decoded data is also zero.

V. Modulation Added

The previously generated Reed-Solomon scheme for coding and decoding is a simple scheme, which does not assume simulation of real physical medium.

In the next implementation of Reed-Solomon coding and decoding process, two main blocks will be added: modulation block, as well as AWGN channel block (Fig. 5). The generated codeword produced by the RS Encoder module, is passed to a digital modulator. The purpose of the modulator is to map group of bits into one symbol to be transmitted, thus increasing the baud rate.

AWGN Channel module is used as a physical medium through which data transmission is conducted. AWGN channel adds white Gaussian noise to the input signal [7]. Fig. 6 shows RS(255,239) code operating with 256-QAM modulation scheme. System simulation parameters are defined as in Table 2.

TABLE II
RS(255,239) System Simulation Parameters

Property	**Value**
Sample time in seconds	1
Codeword length (n)	255
Message length (k)	239
QAM order	256
Bits per symbol	8
Uncoded Eb/No	15

The period of each frame in the model is set to 239 seconds, which corresponds to a sample time of 1 second for every information symbol produced at the output of the Random Integer Generator. The symbol time at the output of the Integer-Input RS Encoder is reduces by a factor of the code rate, because 255 symbols are produced over a frame of 239 symbols.

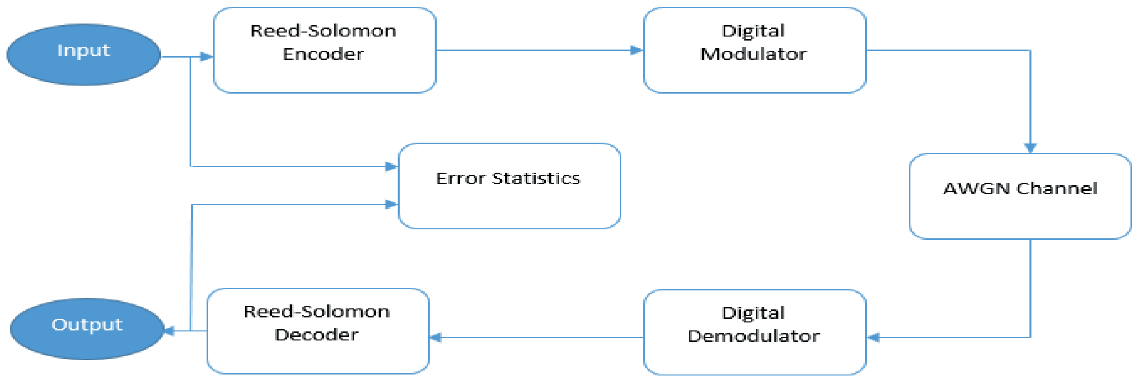

Fig. 5. RS(255,239) main coding and modulation blocks.

The parameters of AWGN channel are defined as in Table 3.

TABLE III
RS(255,239) AWGN CHANNEL PARAMETERS

Field	Description and value
Mode	Signal to noise ratio (Eb/No) [8]
Eb/No (dB)	Ratio of information bit energy to noise power spectral dencity (without channel coding), in decibels
Number of bits per symbol	8
Input signal power	Power of the symbols at the input of the block. This field value is calculated as: Uncoded Eb/No+10*log10(k/n)
Symbol period	The duration of an information channel symbol (without channel coding), in seconds. This field is set to 1

Two bit error rate calculations are performed. The first one is coded BER, which compares the message prior to entering the RS encoder and the message after being decoded by RS decoder. The second one is channel BER, which compares the coded message, after being encoded by the RS encoder, and the received message, before being decoded by the RS decoder.

In Fig. 6 we can see that the RS decoder successfully corrected certain number of errors, which produces no errors in coded BER block, after the message is being decoded. This is shown by the zero value on the plot produced by the Scope module (Fig. 3). The channel BER gives the number of errors generated after the data transmission, but before the decoding process. The channel BER is calculated by the Error Rate Calculation Block (Fig. 6). It can be concluded that all errors produced during transmission were successfully corrected by the RS decoder.

IEEE 802.11ad standard suggest using the shortened version of RS(255,239) as coding technique in Low-Power Single Carrier. Shortening means removing symbols from the information message before entering the encoder, not transmitting them and then re-inserting them at the decoder.

For the shortened version of the RS(255, 239) code, which is RS(224,208), the first 31 terms in information polynomial in (8) are set to zero and only 208 information bytes and 16 parity bytes are transmitted.

In Simulink, shortening the code is incorporated in RS Encoder and RS Decoder blocks. To shorten the RS(255,239) code into RS(224,208) code, the codeword length value, n, needs to be set to 224, and the message length value, k, needs to be set to 208.

VI. CONCLUSION

In this paper, the complete process of implementing RS(255,239) code was described. The system was constructed and simulated using Simulink, showing the

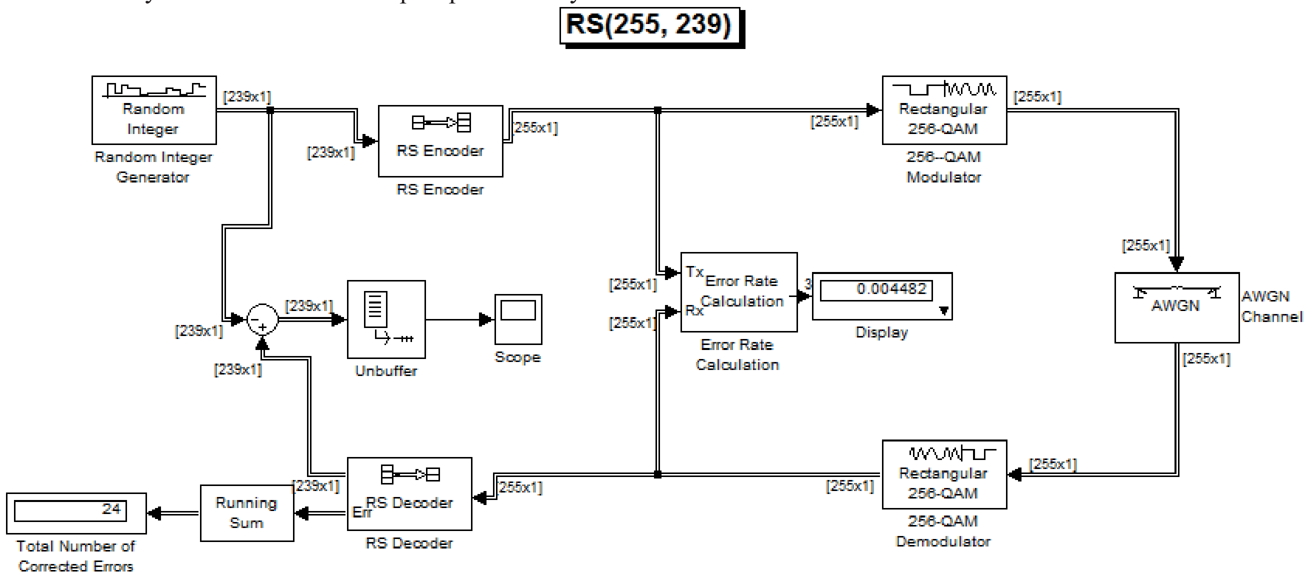

Fig. 6. RS(255,239) coding an modulation process fully implemented in Simulink.

correct implementation of the encoding process. This implementation will play big role in implementing complete IEEE 802.11 ad Low-Power Single Carrier system. Future improvements will be conducted, in terms of research how minimizing the n-k factor will affect on the performance of the code and still maintaining efficient error correcting capabilities.

REFERENCES

[1] W. W. Pwterson, E. J. Weldon, Jr., "Error Correcting Codes," 2nd ed., MIT Press, Cambridge, Massachusetts, 1972.

[2] "Wireless LAN at 60 GHz – IEEE 802.11ad Explained," Application Note, Agilent Technologies.

[3] S. B. Wicker and V. K. Bhg\argava, "Reed-Solomon Codes and Their Application," IEEE Press, 1983 .

[4] F. J. MacWilliams and N. J. A. Sloane, "The Theory of Error Correcting Codes, Part I," North-Holland Publishing Company, Amsterdam, New York, Oxford, 1977.

[5] E. A. Berlekamp, "Algebraic Coding Theory," McGraw-Hill, New York, 1968.

[6] B. Sklar, "Digital Communications: Fundamentals and Application," 2nd ed., Prentice Hall, 2001.

[7] J. P. Odenwalder, "Error Control Coding Handbook," Linkabit Corporation, San Diego, CA, July 15, 1976.

[8] J. Hagenauer and E. Lutz, "Forward Error Correction Coding for Fading Compensation in Mobile Satellite Channels," IEEE JSAC, vol . SAC-5, no. 2, February1987, pp. 215-225.