Design and Implementation of RS (255, 223) Detecting Code in FPGA

Bappaditya Kuila
HIT, Haldia
West Bengal, India

ABSTRACT
Reed-Solomon (RS) codes are commonly used in the digital communication. It has high capability to eliminate both random errors and burst errors. In this work, the encoding of RS(255, 223) code is designed, synthesized, and simulated using Verilog language with the device family of virtex4 & device of xc4vfx12 & compare the result with device family Spartan3E & device XC3S100E. 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. So, from syndrome input-output waveform, it has been checked that whether there is any error in the received codeword or not. RS codes are type of burst error detecting codes which has got many applications due to its burst error detection and correction nature. This code is defined over a Galois Field GF(2^8) and has the capability of correcting up to sixteen short bursts of errors.

Keywords
Reed-Solomon code, Linear Feedback Shift Register, Galois Field, Generator Polynomial, Encoder, Constant Multiplier, Syndrome, Verilog language.

1. INTRODUCTION
Reed-Solomon (RS) code which was discovered by Irving S. Reed and Gustave Solomon in Lincoln Laboratory of MIT, Massachusetts in 1960. It is a kind of multi-Bose-Chaudhuri-Hocquenghem (BCH) code with high error correction capability, which is presently one of the most effective and widely used for error control codes [1]. For the revolution of telecommunication, RS code has large contribution [2]. Specifically, RS codes can be used in computer memory and non-volatile memory applications. They are the most frequently used digital error control codes in the world [3]. The RS encoder algorithm is simpler than RS decoder and the most significant components are multipliers. Although the error correcting capability of RS codes is beyond satisfaction, because of the lack of efficient decoding algorithms they were not largely applied in their early years. W.W. Peterson firstly recognized RS codes as a special class of BCH codes [4]. Compared with other linear block codes, in the same coding efficiency, RS code has strong error correction capability and its error correction performance is close to the theoretical value, particularly on the short yards of medium. Not only RS code can correct the random error, but also it corrects unexpected error [5]. Therefore, it is widely used in deep-space communications systems, data storage systems and digital television transmission [6]. RS code is preferred in terrestrial broadcast channel, because it is a mixed channel which has both random error and burst error. In 1977, in the form of concatenated codes, RS codes were notably applied in the Voyager program [7]. In 1982, with the compact disc, there was the first commercial application in mass-produced consumer products, where two interleaved RS codes are used [8]. Today, RS codes are largely implemented in digital storage devices and digital communication standards, though, by more modern low-density parity-check (LDPC) codes or turbo codes, they are being slowly replaced [9]. For example, RS codes are used in the Digital Video Broadcasting (DVB) standard DVB-S, but LDPC codes are used in its successor DVB-S2. RS code belongs to a family of error-correction algorithms known as BCH [10-13]. To process message data, BCH algorithms use finite fields and to detect errors in the encoded data, they use polynomial structures, called "syndromes," [14]. They can determine the presence of errors and compute the correct values by adding the check symbols to the data block. BCH algorithms have strict control over the number of check symbols [15]. Design of some other RS code like RS (204, 188) & RS (255, 251) in FPGA were performed by H. Zhang (California State University, Northridge) & A. S. Das et. al. respectively [16]. RS code is also a linear and polynomial algorithm as it processes message data as discrete blocks and it is used in modular polynomials. J. Bhaumik et. al. , proposed a programmable RS encoder [17]. The received codeword is entered to RS decoder to be decoded, the decoder first tries to check if this codeword is a valid codeword or not. If it does not, errors occurred during transmission. This part of the decoder processing is called error detection, which is done by syndrome. If errors are detected, the decoder tries to correct this error using error correction part by using different algorithms [18-20].

In this work, the encoding of RS(255, 223) code is designed, synthesized and simulated using Verilog language with the help of 32 constant multipliers and by syndrome's simulation waveform, it has been checked whether the received codeword is error free or not. To get the result of encoder, firstly these multipliers are designed, synthesized and simulated using Verilog language. Before proceeding for the main program of encoder, these results are checked in Matlab code also. In the same way, the syndrome is also designed by using 32 syndrome blocks. In Section 2, RS(255, 223) encoder and syndrome are discussed briefly. Synthesis results and simulation waveforms are given in Section 3. In Section 4, performance comparison of RS(255, 223) encoder is shown. Future work is mentioned in Section 5. The paper is concluded in Section 6.

2. RS (255, 223) ENCODER
The topics, discussed in this Section are the elements of GF(2^8), characteristics of RS(255, 223) code, RS encoder block diagram, the design of RS(255, 223) encoder using Linear Feedback Shift Register (LFSR) and basic idea of syndrome.

2.1 Elements of GF(2^8)
Finite field or Galois field is an algebraic theory raised by French mathematics genius Évariste Galois. Galois fields are

very important in coding theory. The RS codes studied in this paper are based on finite fields.

The elements of RS code discussed in this paper are on the field GF(2^8) = 256. There are 2^8 = 256 elements on GF(2^8), among which 255 elements are non-zero [14]. The primitive polynomial on GF (2^8) is p(x) = $x^8 + x^4 + x^3 + x^2 + 1$. From the primitive polynomial $p(\alpha) = x^8 + x^4 + x^3 + x^2 + 1 = 0$, the elements with order greater than "7" can be derived. The 256 elements on field GF (2^8) are shown in Table 1.

Table 1: Elements of Field GF(2^8)

Power $(\alpha)^i$	Polynomial Form	Binary Form	Decimal Form
0	0	00000000	0
α^0	1	00000001	1
α^1	α	00000010	2
α^2	α^2	00000100	4
α^3	α^3	00001000	8
α^4	α^4	00010000	16
α^5	α^5	00100000	32
α^6	α^6	01000000	64
α^7	α^7	10000000	128
α^8	$\alpha^4 + \alpha^3 + \alpha^2 + 1$	00011101	29
α^9	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha$	00111010	58
α^{10}	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2$	01110100	116
α^{11}	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3$	11101000	232
α^{12}	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 + 1$	11001101	205
α^{13}	$\alpha^7 + \alpha^2 + \alpha + 1$	10000111	135
α^{14}	$\alpha^4 + \alpha + 1$	00010011	19
…	…	…	…
α^{253}	$\alpha^6 + \alpha^2 + \alpha^1 + 1$	01000111	71
α^{254}	$\alpha^7 + \alpha^3 + \alpha^2 + \alpha^1$	10001110	142

2.2 Characteristics of RS(255, 223) code

The characteristics of RS(255, 223) code are discussed in this paper are as below:

Degree of the Polynomial: m = 8
Code Length: n = 255
Information Symbols: k = 223
Parity Check Symbols: r = n – k = 2t = 32
Minimum Distance: dmin = n – k +1 = 2t+1 = 33
Error Correcting Capability: t =16
Code Rate = Code Efficiency = k/n = 223/255 = 0.875

However, each symbol is represented by eight binary digits or one byte. Also, each data block contains 223 information symbols. This code is capable of correcting up to sixteen short burst errors of one byte or any burst error combination of up to a total length of eight bytes, providing that they only affect a maximum of sixteen individual symbols [15].

2.3 Construction of GF(2^8)

The elements of GF(2^8) are generated by primitive polynomial of degree 8.

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

Let α be the primitive element in GF(2^8) and the root of p(X)
Then,
p(x) = $X^8 + X^4 + X^3 + X^2 + 1 = 0$
Or
$X^8 = X^4 + X^3 + X^2 + 1$

So, the elements can be represented in an 8-tuple with 8 components being 0 or 1 and represent code word [17]. The zero element of GF(2^8) appears as an all zero 8-tuple.

Also, if α is a primitive element in GF(2^m), then the root of p(x) is only the first thirty-two powers of α and are the roots of the generator polynomial. Meanwhile, the generator polynomial for (255, 223) code is given by:

$$g(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)(x+\alpha^5)(x+\alpha^6)(x+\alpha^7)(x+\alpha^8)(x+\alpha^9)(x+\alpha^{10})(x+\alpha^{11})(x+\alpha^{12})(x+\alpha^{13})(x+\alpha^{14})(x+\alpha^{15})(x+\alpha^{16})(x+\alpha^{17})(x+\alpha^{18})(x+\alpha^{19})(x+\alpha^{20})(x+\alpha^{21})(x+\alpha^{22})(x+\alpha^{23})(x+\alpha^{24})(x+\alpha^{25})(x+\alpha^{26})(x+\alpha^{27})(x+\alpha^{28})(x+\alpha^{29})(x+\alpha^{30})(x+\alpha^{31})(x+\alpha^{32})$$

g(x) = $45 + 216x + 239x^2 + 24x^3 + 253x^4 + 104x^5 + 27x^6 + 40x^7 + 107x^8 + 50x^9 + 163x^{10} + 210x^{11} + 227x^{12} + 134x^{13} + 224x^{14} + 158x^{15} + 119x^{16} + 13x^{17} + 158x^{18} + x^{19} + 238x^{20} + 164x^{21} + 82x^{22} + 43x^{23} + 15x^{24} + 232x^{25} + 246x^{26} + 142x^{27} + 50x^{28} + 189x^{29} + 29x^{30} + 232x^{31} + x^{32}$

Therefore, the coefficients of g(x) used in the encoder multiplication are:

g_0= 45, g_1= 216, g_2 = 239, g_3 = 24, g_4 = 253, g_5= 104, g_6 = 27, g_7 = 40, g_8 = 107, g_9 = 50, g_{10} = 163, g_{11} = 210, g_{12} = 227, g_{13}= 134, g_{14}= 224, g_{15}=158, g_{16}= 119, g_{17}= 13, g_{18}= 158, g_{19}=1, g_{20}= 238, g_{21}= 164, g_{22}= 82, g_{23}=43, g_{24}= 15, g_{25}= 232, g_{26}= 246, g_{27}= 142, g_{28}=50, g_{29}=189, g_{30}=29, g_{31}= 232, g_{32}= 1

2.4 Encoder Architecture

The block diagram of RS encoder is shown in Figure 1 [18].

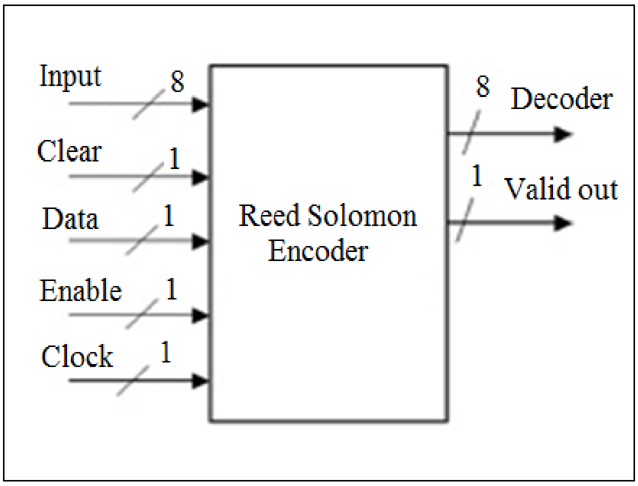

Figure 1: RS Encoder Block Diagram

Encoders are designed as feedback shift register, Figure 2. The data massage blocks of 223 symbols shift sequentially as an input to the encoder and when the last message symbol is loaded, the feedback register contains the thirty-two parity check symbols. These symbols will then be shifted out following the 223 information symbols to generate a code word of 255 symbols as an output of the encoder [20].

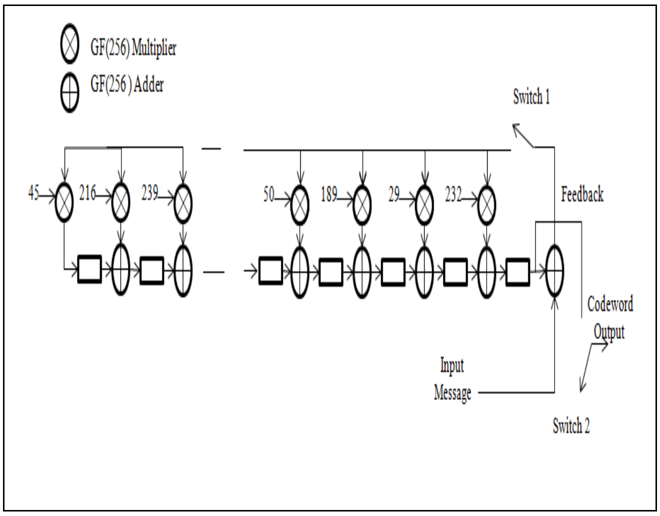

Figure 2: RS(255, 223) Encoder using LFSR

2.5 Syndrome

Syndrome is utilized to determine whether an error happened in the transmission. If value of the syndrome is 0, there is no error in the transmission and the received sequence is the code word; while if syndrome value is non zero, then there is error, hence error correction is needed. Syndrome values are only dependent on the error pattern. It checks if there is any error in the received codeword or not.

$$\mathbf{r(X) = c(X) + e(X)}$$

Where, code word = c(x), error pattern = e(x), received signal = r(x)

There are 32 syndrome blocks which has helped us to get the input and output waveform of syndrome. In Table 2, the values of 32 syndrome blocks are shown. Firstly these 32 syndrome blocks are synthesized and simulated using Verilog code with the device family of virtex4 & device of xc4vfx12. The syndrome blocks' simulation result is checked by Matlab code. Then the syndrome is synthesized and simulated with the help of these syndrome blocks.

Table 2: Syndrome Blocks

SYNDROME BLOCKS	
$S_1 = 2$	$S_{17} = 152$
$S_2 = 4$	$S_{18} = 45$
$S_3 = 8$	$S_{19} = 90$
$S_4 = 16$	$S_{20} = 180$
$S_5 = 32$	$S_{21} = 117$
$S_6 = 64$	$S_{22} = 234$
$S_7 = 128$	$S_{23} = 201$
$S_8 = 29$	$S_{24} = 143$
$S_9 = 58$	$S_{25} = 3$
$S_{10} = 116$	$S_{26} = 6$
$S_{11} = 232$	$S_{27} = 12$
$S_{12} = 205$	$S_{28} = 24$
$S_{13} = 135$	$S_{29} = 48$
$S_{14} = 19$	$S_{30} = 96$
$S_{15} = 38$	$S_{31} = 192$
$S_{16} = 76$	$S_{32} = 157$

3. SYNTHESIS RESULTS AND SIMULATION WAVEFORMS

In this section, synthesis results and simulation waveforms of 32 coefficients, RS(255, 223) encoder, 32 syndrome blocks and syndrome have been elaborated.

In Table 3, synthesis result of 32 coefficients used in RS(255, 223) encoder is shown. These results are got in Verilog language using device family virtex4 and device xc4vfx12.

Table 3: Synthesis Result for Coefficients of Generator Polynomial g(x)

Coefficients	Number of Slices	Number of 4 Input LUTs	Number of Bounded IOBs	Delay (ns)
1	…	…	16	3.562
13	6	10	16	4.95
15	7	12	16	4.97
24	5	8	16	4.288
27	6	10	16	4.949
29	5	9	16	4.949
40	6	10	16	4.957
43	5	8	16	4.281
45	6	10	16	4.957
50	6	10	16	4.949
104	5	9	16	4.971
107	8	14	16	4.949

119	6	11	16	4.971
134	4	7	16	4.283
142	2	3	16	4.281
158	4	7	16	4.288
163	7	13	16	4.957
164	5	8	16	4.282
189	6	11	16	4.963
210	7	12	16	4.961
216	3	6	16	4.288
224	6	11	16	4.971
227	6	11	16	4.971
232	6	10	16	4.957
238	5	9	16	4.971
239	7	12	16	4.957
246	8	14	16	4.971
253	7	13	16	4.895

After getting these synthesis result, all the 32 coefficients are simulated. The simulation waveform in Figure 3, for coefficient g_0 is shown. All the results are checked using Matlab code.

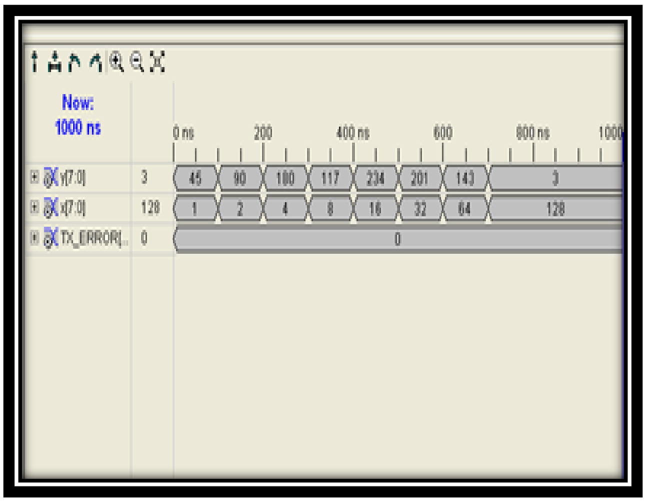

Figure 3: Simulation Waveform for Coefficient g_0 = 45

Using Matlab code, we can Verify the Simulation Result for Coefficient g_0 = 45:

This code is defined over Galois Field GF(2^8) and the primitive polynomial is $X^8 + X^4 + X^3 + X^2 + 1$ (285 decimal). For the coefficient g_0 = 45, if we multiply it by 1, 2, 4, 8, 16, 32, 64, 128. The result will come 45, 90, 180, 117, 234, 201, 143, 3.

In Table 4, synthesis result of RS(255, 223) encoder is shown. These results are got in Verilog language using device family virtex4 and device xc4vfx12.

Table 4: Synthesis Result of RS(255, 223) Encoder

Number of Slices	Number of Slice Flip Flops	Number of 4 Input LUTS	Number of IOS	Number of Bounded IOBS	Number of GCLKS	Delay (ns)
239	256	455	22	22	1	3.014

After getting the synthesis result, the input and output waveform for RS(255, 223) encoder is shown in Figure 4 for the input 1 in all 223 input message signal.

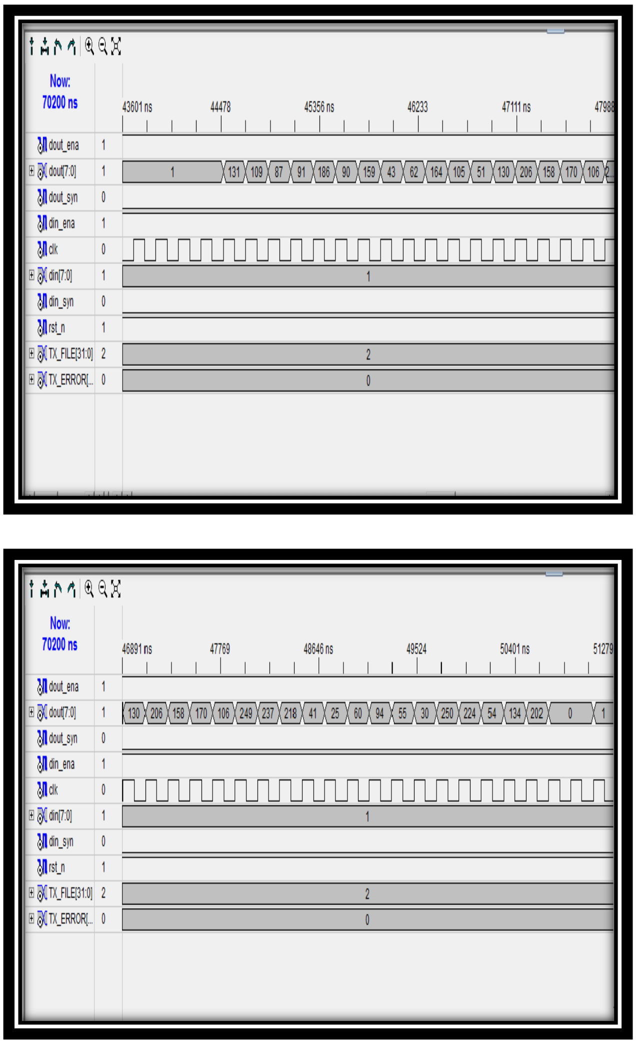

Figure 4: The parity bits obtained when input is '1' for all 223 input message signal (The parity bits are respectively 131, 109, 87, 91,………………, 54, 134, 202. The next output '0' shows that encoding of code is completed)

In Table 5, synthesis result of syndrome blocks is shown. These results are got in Verilog language using device family virtex4 and device xc4vfx12.

Table 5: Synthesis Result of Syndrome Blocks

Syndrome Blocks	Number of Slices (out of 5472)	Number of 4 Input LUTs (out of 10944)	Delay (ns)
S_1 = 2	2	3	4.868
S_2 = 4	2	4	4.946
S_3 = 8	3	5	4.986
S_4 = 16	3	6	4.986
S_5 = 32	4	7	4.986
S_6 = 64	4	8	4.993
S_7 = 128	4	8	4.993
S_8 = 29	5	9	4.949
S_9 = 58	5	9	5.277
S_{10} = 116	5	9	5.277
S_{11} = 232	6	10	4.957
S_{12} = 205	5	9	5.582
S_{13} = 135	5	9	5.284
S_{14} = 19	5	9	5.419
S_{15} = 38	5	9	5.541
S_{16} = 76	5	9	5.569
S_{17} = 152	5	9	5.627
S_{18} = 45	6	10	4.957
S_{19} = 90	5	9	5.284
S_{20} = 180	4	8	5.543
S_{21} = 117	5	9	5.294
S_{22} = 234	4	8	5.542
S_{23} = 201	4	8	4.986
S_{24} = 143	4	8	4.953
S_{25} = 3	4	8	4.949
S_{26} = 6	4	8	4.989
S_{27} = 12	4	8	4.993
S_{28} = 24	5	8	4.288
S_{29} = 48	4	8	5.543
S_{30} = 96	4	8	5.543
S_{31} = 192	4	8	5.620
S_{32} = 157	4	8	5.660

After getting the synthesis result, the input and output waveform for syndrome block S_3 is shown in Figure 5. All the syndrome blocks are simulated using Verilog language and the results are checked by Matlab code.

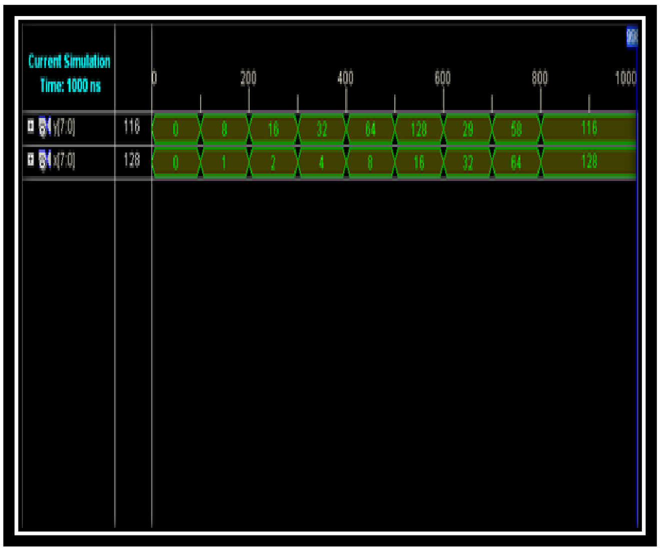

Figure 5: Simulation Waveform of Syndrome Block S_3= 8

Using Matlab code, we can Verify the Simulation Result for Coefficient S_3 = 8:

This code is defined over Galois Field GF(2^8) and the primitive polynomial is $X^8 + X^4 + X^3 + X^2 + 1$ (285 decimal). For the coefficient S_3 = 8, if we multiply it by 1, 2, 4, 8, 16, 32, 64, 128. The result will come 8, 16, 32, 64, 128, 29, 58, 116.

In Table 6, synthesis result of syndrome is shown. The result is got in Verilog language using device family virtex4 and device xc4vfx12.

Table 6: Synthesis Result of Syndrome

Number of Slices	Number of Slice Flip Flops	Number of 4 Input LUTS	Number of GCLKS	Delay (ns)
408	256	789	1	2.131

After getting the synthesis result, the input and output waveform for syndrome is shown in Figure 6.

In the input of this waveform it is seen that, 32 parity bits are given, which are got from simulation result of RS(255, 223) Encoder in Figure 4, when input is ‘1’ for all 223 input message signal. In the simulation result, syndrome is non zero, so there is error in the received codeword, hence error correction is needed.

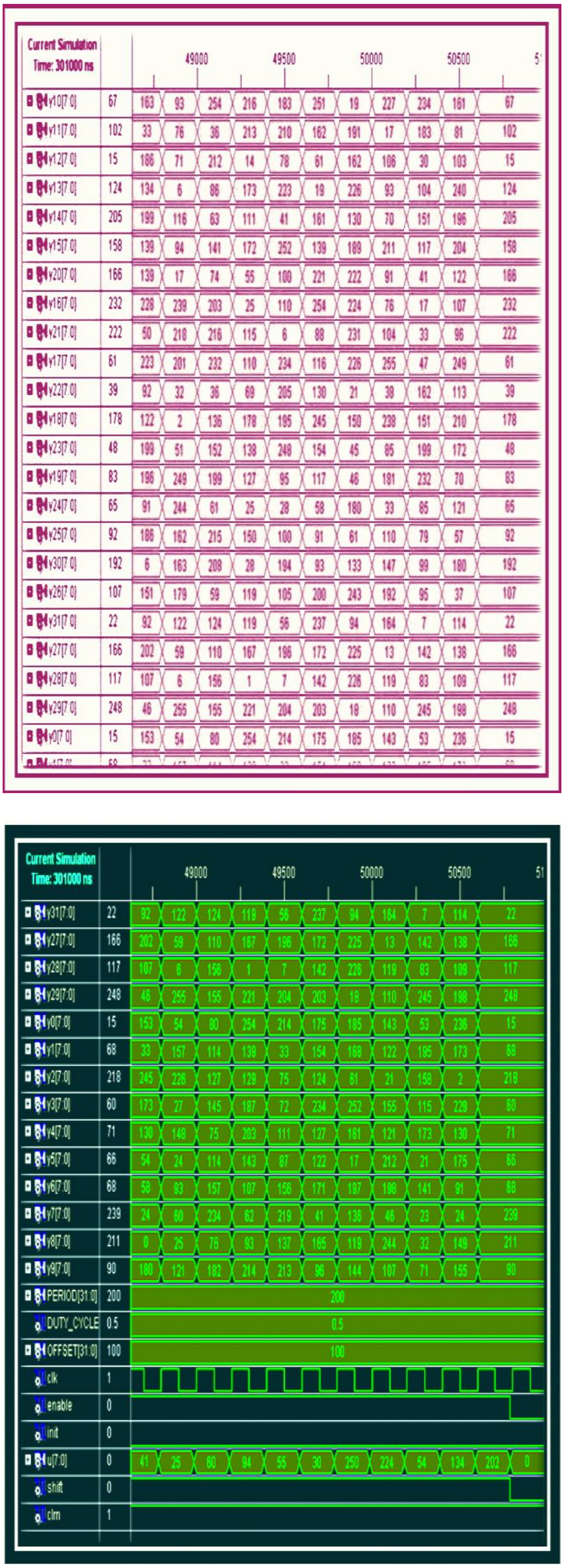

Figure 6: Simulation Waveform of Syndrome

4. PERFORMANCE COMPARISON

In Table 4, the synthesis result of RS(255, 223) encoder by using the device family of Virtex4 & device of Xc4vfx12 is shown. It is compared with device family of Spartan3E & device of XC3S100E. The performance comparison between these two is listed in Table 4, where it is seen that number of Slice Flip-Flops, IOBS, bounded IOBS & GCLKS are same for both the cases. But for device family Virtex4, number of Slices is needed more than device family Spartan3E, whereas Spartan3E's delay is greater than the device family Virtex4.

Table 7: Performance Comparison of Synthesis Result for RS(255, 223) encoder

Device Used	Number of Slices	Number of Slice Flip Flops	Number of 4 Input LUTS	Number of IOS	Number of Bounded IOBS	Number of GCLKS	Delay (ns)
Virtex4 & device: Xc4vfx12	239	256	455	22	22	1	3.014
Spartan 3E & device: XC3S100E	238	256	453	22	22	1	6.124

5. FUTURE WORK

In this work, the synthesis result with simulation waveform of RS(255, 223) encoder and syndrome are shown. During the transfer of message, the data might get corrupted due to lots of disturbances in the communication channel. In the syndrome's input-output waveform, it is seen that the value of the syndrome is non zero. So there is error in the transmitted codeword. By correcting these errors, we can recover the actual codeword. So the future work will be,

A. Determine the roots of which are related to the error locations using Chien Search

B. Calculate the values of the error evaluator using Forney Algorithm

C. Recover the corrected codeword by adding E(x) with R(x)

6. CONCLUSION

The communication channel in modern digital and data storage systems requires error detecting and correcting codes to correct the errors that occur during the transmission of data. It can be also implemented on Visual Sensor Network (VSN), Deep-Space communication and Digital μ-wave radio. In this paper, RS encoding, system specification of RS (255, 223) encoder with its architecture & design using LFSR are discussed. The co-efficients of generator polynomial used in the encoder multiplication are mentioned. These terms are simulated using Verilog code & whether the simulation results are right or wrong, are tested by Matlab code which has helped to design encoder. With the help of these co-efficient terms, the simulation waveform of RS(255, 223) encoder is got by using the Verilog code and performance comparison of RS(255, 223) encoder using a different device is shown. Lastly syndrome is simulated by using the syndrome blocks which is also shown in this paper. As the received codeword is erroneous, so error correction is necessary. So, in this paper, it is detected whether the received code word is error free or not.

7. ACKNOWLEDGEMENTS

I am very thankful to Mr. Jagannath Samanta who has contributed towards the development of this work.

8. REFERENCES

[1] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics, 8:300-304, 1960.

[2] ETSI, 1997. Digital broadcasting systems for television, sound and data services; Framing structure, channel coding and modulation for digital terrestrial television. European Telecommunication Standard ETS 300 744.

[3] S.B. Wicker. Error Control Systems for Digital Communication and Storage. Prentice-Hall, Englewood Clips, NJ, 1995.

[4] G. D. Forney, Jr. On decoding BCH codes. IEEE Transactions on Information Theory, IT-11:549-557, 1965.

[5] S. Liu and Jr. D. J. Costello. Error Control Coding: Fundamentals and Applications. Prentice-Hall, Englewood Clips, NJ, 1983.

[6] Akyildiz, I.F., T. Melodia, and K.R. Chowdury, Wireless multimedia sensor networks: A survey. IEEE Wireless Communications. 2007, 14(6): p. 32-39.

[7] H. Y. Hsu and A. Y. Wu, "VLSI Design of a Reconfigurable Multimode Reed-Solomon Codec for High Speed Communication Systems," in IEEE Asia-Pacific Conference on ASIC, 2002, pp. 359-362.

[8] K.A.S. Immink, \Reed Solomon codes and the compact disc," in Reed-Solomon Codes and Their Applications, eds. S.B. Wicker and V.K. Bhargava. New York: IEEE Press, 1994, pp. 41-59.

[9] Stephen B. Wicker and Vijay K. Bhargava, "Reed-Solomon codes and their applications", IEEE Press, New Jersey, 1994.

[10] J. L. Massey. Shift register synthesis and BCH decoding. IEEE Trans-actions on Information Theory, pages 122 - 127, January 1969.

[11] Sklar B, "Digital Communication: Fundamentals and Applications", Second Edition, Prentice-Hal, 2001.

[12] Pretzel, O. 1992.Error-correcting codes and finite fields. Clarendon Press, Oxford, 1992.

[13] R. T. Chien. Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. IEEE Transactions on Information Theory, IT-10:357 - 363, October 1964.

[14] Y. R. Shayan and T. Le-Ngoc, "Decoding Reed-Solomon codes generated by any generator polynomial," Electronics Letters, vol. 25, no. 18, Aug. 1989, pp. 1223-1224.

[15] T. Yaghoobian and I.F. Blake "Reed Solomon and Algebraic Geometry Codes," in Reed-Solomon Codes and Their Applications, eds. S.B. Wicker and V.K. Bhargava. New York: IEEE Press, 1994, pp. 292-314.

[16] Anindya Sundar Das, Satyajit Das and Jaydeb Bhaumik, Design of RS (255, 251) Encoder and Decoder in FPGA . International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January.

[17] Jaydeb Bhaumik, Anindya Sundar Das and Jagannath Samanta, Architecture for Programmable Generator Polynomial Based Reed-Solomon Encoder and Decoder. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January

[18] J. Bhaumik and D. Roy Chowdhury, "An Integrated ECC-MAC Based on RS Code," Transactions on Computational Science, vol. IV, LNCS5430, Apr. 2009, pp. 117-135.

[19] Syed Shahzad Sha, Saqib Yaqub, and Faisal Suleman, " Self-correcting codes conquer noise Part 2: Reed-Solomon codec's", EDN, pp. 107-120, March 2001.

[20] J. Jittawutipoka and J. Ngarmnil, "Low complexity Reed-Solomon encoder using globally optimized finite field multipliers", TENCON 2004, 2004 IEEE Region 10 Conference, vol. D, 21-24 Nov. 2004, pp. 423-426, doi: 10.1109/TENCON.2004.1414960.