Forward Error Correction For Gigabit Automotive Ethernet using RS(450,406) Encoder

Akhilesh Yadav, Poonam Jindal, Devaraju Basappa, Mahendra Prakashaiah

Abstract*: Error correction and detection during data transmission is a major issue. For resolving this, many error correction techniques are available. The Reed-Solomon coding is the most powerful forward error correction technique used in Gigabit Automotive Ethernet to compact channel noise during data transmission. The car becomes more smarter day by day and more new advanced electronics is being used in-vehicle. Gigabit Automotive Ethernet(1000BASE-T1) provide fast bandwidth for many kinds of applications and connect different functional parts in the car. The Reed Solomon(RS) coding is the powerful forward error correction(FEC) technique used in 1000BASE-T1 Automotive Ethernet. RS(450,406) coding is also known as shortened Reed Solomon codes. The Reed Solomon(RS) codes are generally used in communication system due to its ability of correcting both random and burst errors. Reed Solomon codes are no-binary systematic linear block codes. RS coding is widely used in high speed communication system. This RS code is implemented using Galois field(GF). The Automotive Ethernet is encoded using RS(450,406) codes through GF(512) for FEC. This RS codes can corrects the error up to t=22 symbol, while other encoding techniques corrects the error in t bits. In this paper we implemented the RS(Reed Solomon) code in Cadence ncsim Verilog software and used Cadence Simvision for showing timing diagrams. This RS code uses 9-bit based shortened (450,406) code.

Keywords* : Automotive Ethernet, Cadence, Galois Field, Generator polynomial, ncsim, Reed Solomon, RS encoder, Primitive elements, Primitive polynomial, Simvision, Verilog.

I. INTRODUCTION

Due to the higher data rates required in Automotive Ethernet(1000BASE-T1), Reed Solomon codes[1] are used for error correction and detection. Reed Solomon codes can correct contiguous error (burst error) and random error. In forward error correction(FEC)[2] techniques redundant bits are added into the original message, so errors can be easily corrected. Reed Solomon(RS) code is the subclass of standard q-array BCH code, mostly used in storage systems and digital communication system for controlling the error[3]. RS(n, k) is the representation of the Reed Solomon code, where k is the message symbols and n is the total codeword symbol. RS provides better efficiency and less complexity to other FEC techniques. The main difference between RS and BCH code is that, RS code corrects t symbols while BCH code corrects t bits error.

Revised Manuscript Received on December 12, 2019.
***** Correspondence Author
Akhilesh Yadav, NIT Kurukshetra, Email: yadav18091995@gmail.com

RS(450,406) can corrects up to 22 symbol errors. Reed Solomon(RS) codes are based on Galois Field(GF) [4],[5],[6]Arithmetic. Forward error correction (FEC) is the error controlling method by adding redundant bits in the original message as shown in Fig. 1, where k message symbols are taken as input and redundant symbols are added to make n symbols codeword. FEC encoder blocks includes different encoding techniques, we are using shortened RS(450,406) encoder.

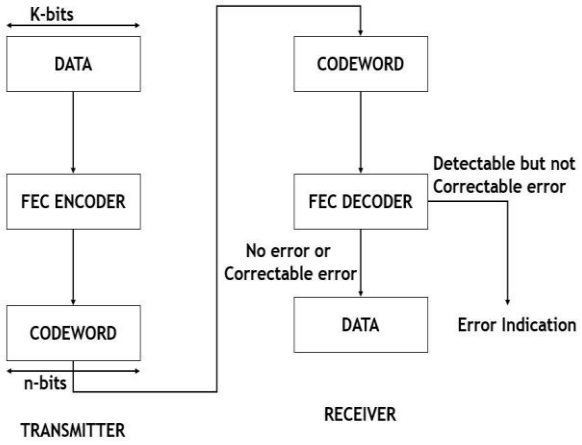


Fig. 1. Forward error correction process.

DSP chips progressively support forward error correction (FEC) in many storage system. Gigabit Ethernet endeavor directly stated FEC in DSP chips.

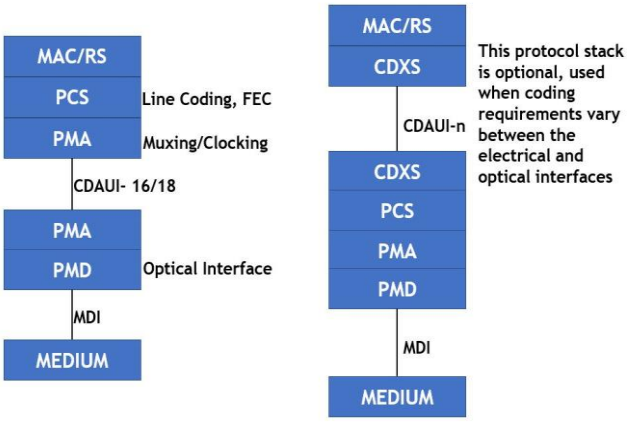


Fig. 2. Some protocol stack with Forward error correction.

International Telecommunication Union (ITU) legally made a Reed Solomon (RS) FEC in optical transport network, FEC also used in 10Gbps network elongation in Ethernet and also in MAC, PHY[7]. IEEE 802.3 standard requires higher speed and more powerful error correction and detection capability, so we used RS(450,406) code. Fig. 2 denotes the forward error correction protocol stack. This paper starts with basic idea about Reed Solomon code, Galois Filed(GF) arithmetic in section II. In section III we described about RS(450,406) encoding process, Implementation and results are shown in section IV and finally.

II. REED SOLOMON CODE

Reed Solomon codes are systematic linear block codes and it is the part of BCH code. RS code is widely used in communication and storage system due to their simplicity and complexity. Reed Solomon codes are represented as RS(n, k) where n is the length of final codeword and k is the length of original message and it can correct the error up to t symbols.

$$2t = n - k$$
$$t = \frac{(n-k)}{2} \quad (1)$$

Minimum distance:- $dmin = 2t + 1$
Dimension: $k = q - 1 - 2t$
Block length $n = q - 1$
Fig. 3 shows a systematic Reed Solomon(RS) codeword. For RS(450,406) code, n = block length= 450, k = 406 , 2t = 44 and t=22(max number of error to be corrected). Now we can say that $dmin$ is greater than the parity bit symbol.

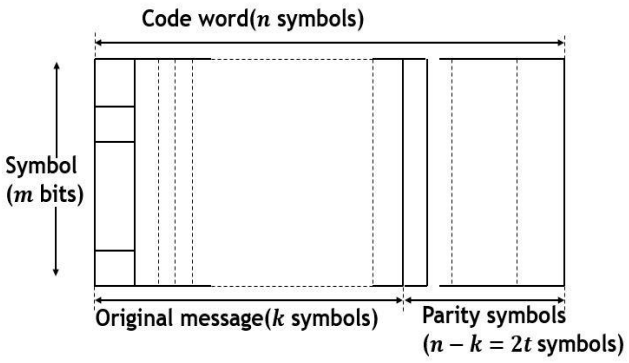


Fig. 3. Reed Solomon codeword.

A. *Galois Field(GF)*

Reed Solomon code is the based on the Galois Field GF(512). Galois field having elements that are generated due to the primitive elements and if the length of the symbol is m, then length of elements in GF will be 2^m. In the Galois field primitive elements α is generated with the help of the primitive polynomials. The number of primitive polynomial is given in the Table I.
For RS(450,406) we used primitive polynomial of GF(2^9)[8].

$$p(x) = 1 + X^4 + X^9 \quad (2)$$

Primitive elements α takes value 0, α^0, α^1, α^2,…, α^{N-1}, where value of N= 2^m-1. Primitive polynomial for GF(512), $p(x)$, and we know that α will be the root of $p(x)$, so we can write:

$$p(\alpha) = 0$$
$$p(\alpha) = 1 + \alpha^4 + \alpha^9 = 0 \quad (3)$$
$$1 + \alpha^4 = \alpha^9$$

TABLE- I: LIST OF GF PRIMITIVE POLTNOMIALS

m	Primitive polynomials
3	$1 + X + X^3$
4	$1 + X + X^4$
5	$1 + X^2 + X^5$
6	$1 + X + X^6$
7	$1 + X^3 + X^7$
8	$1 + X^2 + X^3 + X^4 + X^8$
9	$1 + X^4 + X^9$
10	$1 + X^3 + X^{10}$
11	$1 + X^2 + X^{11}$
12	$1 + X + X^4 + X^6 + X^{12}$
13	$1 + X + X^3 + X^4 + X^{13}$
14	$1 + X + X^6 + X^{10} + X^{14}$
15	$1 + X + X^{15}$

TABLE- II: REPRESENTATION OF PRIMITIVE ELEMENTS FOR GF(2^9)
BY $p(x) = 1 + X^4 + X^9$

Index form	9-Tuple representation $(\alpha^8\alpha^7\alpha^6\alpha^5\alpha^4\alpha^3\alpha^2\alpha^1\alpha^0)$	Decimal Represent at-ion
0	(000000000)	0
α^0	(000000001)	1
α^1	(000000010)	2
α^2	(000000100)	4
α^3	(000001000)	8
α^4	(000010000)	16
α^5	(000100000)	32
α^6	(001000000)	64
α^7	(010000000)	128
α^8	(100000000)	256
α^9	(000010001)	17
α^{10}	(000100010)	34
α^{11}	(001000100)	68
α^{12}	(010001000)	136
α^{13}	(100010000)	272
α^{14}	(000110001)	49
	.	
	.	
	.	
α^{509}	(010000100)	132
α^{510}	(100001000)	264

where α^9 is the coefficient of GF(512) and it can be written in the form of 9-bit binary number like as 0000010001, we can calculates other GF primitive elements as given in Table II.
Log of Galois field is given as:

If $a^x = y$ then the value of $log(y) = x$
And the inverse log of Galois field is defined as: If $a^x = y$ then the value of $log^{-1}(x) = y$
By using the definition of log-inverse and log, we can easily define the division and multiplication for Galois field elements as:

$$0.a = 0$$
$$a.b = log^{-1}((log(a) + log(b))\ modulo\ 511)$$
$$a/b = log^{-1}((log(a) - log(b))\ modulo\ 511)$$

In GF(2) the polynomial satisfy the condition as listed below.

1. Commutative:
$$a(X) + b(X) = b(X) + a(X),$$
$$a(X).b(X) = b(X).a(X).$$
2. Associative:
$$a(X) + [b(X) + c(X)] = [a(X) + b(X)] + c(X),$$
$$a(X).[b(X).c(X)] = [a(X).b(X)].c(X).$$
3. Distributive:
$$a(X).[b(X) + c(X)] = [a(X).b(X)] + a(X).c(X).$$

Subtraction and Addition operations are same in Galois field elements[9],[10] and it is defined as:

$$a_{m-1}a_{m-2} \dots a_1a_0 + b_{m-1}b_{m-2} \dots b_1b_0 = a_{m-1}a_{m-2} \dots a_1a_0 - b_{m-1}b_{m-2} \dots b_1b_0$$

Where $c_n = a_n xor\ b_n$

The pseudocode of algorithm for generating primitive elements for GF(512) is given below.

Algorithm I

Initialization:

$PP = 0000010001, temp = 0, pm(i)\ for\ i = 0,1,2,3 \dots \dots 2^9 - 1.$

for $i = 0$ **step 1 until** $2^9 - 1$ **do**
begin
Step 1 **if** $i == 0$
then
begin
$temp = (10'b0000000001);$
$pm[i] = temp[8\ to\ 0];$
end
else
begin
$temp = temp << 1;$
if $temp = 1$
begin
$temp = temp + (PP);$
end
$pm[i] = temp[8\ to\ 0];$
end
end

The condition temp=1 is true if and only if the most significant bit of the temp is equal to one, temp is the any temporary variable.

B. *GF Multiplication And Division*

Galois field multiplication is slightly different from normal multiplication, we are explaining through an example.

If $a(X) = X^5 + X^2 + X$ and
$b(X) = X^7 + X^4 + X^3 + X^2 + X$

In GF(2^9) the irreducible polynomial is $X^9 + X^4 + 1$, so

$$\begin{aligned} a(X) \otimes b(X) &= (X^5 + X^2 + X) \otimes (X^7 + X^4 + X^3 + X^2 + X) \\ &= X^5(X^7 + X^4 + X^3 + X^2 + X) + \\ &\quad X^2(X^7 + X^4 + X^3 + X^2 + X) \\ &\quad + X(X^7 + X^4 + X^3 + X^2 + X) \\ &= X^{12} + X^7 + X^2 \end{aligned} \quad (4)$$

If the degree of the multiplication output polynomial is greater than or equal to the irreducible polynomial then divides the resulting polynomial by the irreducible polynomial. So after the division output will be $X^3 + X^2$. The algorithm of GF multiplication[11] and division is given below.

Algorithm II

Initialization:
Parameter N
Parameter M
$f = 0, c = 0$
Input a [N-1 to 0], Input b[M-1 to 0], output c[M+N-2 to0]
for $i = 0$ **step1 until** $N - 1$ **do**
begin
for $j = 0$ **step1 until** $M - 1$ **do**
begin
Step1: if($a[i] == 1\ and\ b[i] == 1$)
begin
$c[i + j] = \sim c[i + j];$
end
end
end

Another way to calculate the multiplication by multiply individual bits of b to a and use shift operator after individual multiplication. After the multiplication if the degree is greater than or equal to the degree of the primitive polynomial then we divides the multiplication output by the GF(2^9) primitive polynomial so GF division algorithm is given Algorithm III.

Algorithm III

Initialization:
Parameter N
Parameter M
counter= length of the $c, j = 9$
Input: PP=000000000000010001
Input: c(output of the multiplication)
output f[N-1 to 0]
for $j = 0$ **step1 until** $M - 1$ **do**
begin
step1: if ($c[i] == 0$)
begin
counter = counter -1;
f=c;
end
else
begin
if ($i > j$)
begin

```
        c = c xor of ( pp << (i − j));
        f=c;
      end
      else if ( i == j)
       begin
              f = c xor of pp ;
       end
        else
        begin
           f = c;
        end
end
```

III. RS(450, 406) ENCODING PROCESS

In (1000BASE-T1) stream is Reed Solomon (RS) [12],[13] encoded by using the RS(450, 406) code over the Galois Field GF(512) for FEC (forward error correction) and it can corrects the errors up to t=22 symbols per RS block code. Primitive polynomial for Galois field GF (512) is $p(x) = 1 + X^4 + X^9$. This polynomial satisfy $p(\alpha) = 0$ and the value of α taken in Hexadecimal.

A. *Generation Of Generator Polynomial*

The RS (450, 406) code used the generator polynomial by the encoder is given below.

$$g(x) = \prod_{i=0}^{2t-1}(x + \alpha^i) \qquad (5)$$

$$g(x) = (x+\alpha^0)\ (x+\alpha^1)\ (x+\alpha^2)\ \ldots\ldots\ (x+\alpha^{43})$$

$$\begin{aligned} g(x) = {} & x^{44} + \alpha^{217}x^{43} + \alpha^{328}x^{42} + \alpha^{11}x^{41} \\ & +\alpha^{57}x^{40} + \alpha^{33}x^{39} + \alpha^{434}x^{38} \\ & +\alpha^{193}x^{37} + \alpha^{46}x^{36} + \alpha^{66}x^{35} \\ & +\alpha^{314}x^{34} + \alpha^{25}x^{33} + \alpha^{70}x^{32} \\ & +\alpha^{16}x^{31} + \alpha^{381}x^{30} + \alpha^{10}x^{29} \\ & +\alpha^{452}x^{28} + \alpha^{395}x^{27} + \alpha^{35}x^{26} \\ & +\alpha^{419}x^{25} + \alpha^{510}x^{24} + \alpha^{7}x^{23} \\ & +\alpha^{447}x^{22} + \alpha^{50}x^{21} + \alpha^{85}x^{20} \\ & +\alpha^{37}x^{19} + \alpha^{207}x^{18} + \alpha^{99}x^{17} \\ & +\alpha^{199}x^{16} + \alpha^{311}x^{15} + \alpha^{214}x^{14} \\ & +\alpha^{403}x^{13} + \alpha^{500}x^{12} + \alpha^{498}x^{11} \\ & +\alpha^{319}x^{10} + \alpha^{114}x^{9} + \alpha^{137}x^{8} \\ & +\alpha^{327}x^{7} + \alpha^{100}x^{6} + \alpha^{253}x^{5} \\ & +\alpha^{320}x^{4} + \alpha^{317}x^{3} + \alpha^{166}x^{2} \\ & +\alpha^{98}x^{1} + \alpha^{435} \end{aligned}$$

B. *Reed Solomon Encoder Design*

The inputs to RS(450, 406) encoder having 406 message of every 9-bit symbol, starts from first m_{405} symbol and with the message symbol m_0. And each symbol has 9-bit output $m_{i,0}, m_{i,1}, m_{i,2}, \ldots, m_{i,8}$, where $m_{i,0}$ is the first bit and $m_{i,8}$ is the last bit and it is mapped to Reed Solomon symbol $m_i = m_{i,0}, m_{i,1}, \ldots. m_{i,8}$ with the representation of the field $m_{i,8}\alpha^8 + m_{i,7}\alpha^7 + \cdots + m_{i,1}\alpha^1 + m_{i,0}$. So the input message polynomial to the encoder circuit is described as:

$$\begin{aligned} m(x) = {} & m_{405}x^{405} + m_{404}x^{404} + m_{403}x^{403} \\ & + m_{402}x^{402} + \cdots + m_1x^1 + m_0 \end{aligned} \qquad (6)$$

In Reed Solomon encoder the input message polynomial $m(x)$ is first shifted by $x^{n-k}(x^{44})$ or just multiplied by $x^{n-k}(x^{44})$ to the message polynomial, Then the new shifted message is divided by the $g(x)$ which is the generator polynomial for RS(450, 406). And remainder is described by $r(x)$:

$$r(x) = r_{43}x^{43} + r_{42}x^{42} + r_{41}x^{41} + \cdots + r_1x^1 + r_0 \qquad (7)$$

$$\begin{aligned} c(x) = {} & m_{405}x^{449} + m_{404}x^{448} + m_{403}x^{447} + \cdots + m_1x^{45} + \\ & m_0x^{44} + r_{43}x^{43} + r_{43}x^{43} + r_{42}x^{42} + \cdots + r_1x^1 + r_0 \end{aligned} \qquad (8)$$

The output message coefficient from the Reed Solomon encoder is $m_{405}m_{404}m_{403}\ldots m_1m_0r_{43}r_{42}r_{41}\ldots r_1r_0$ and the order is left to right.

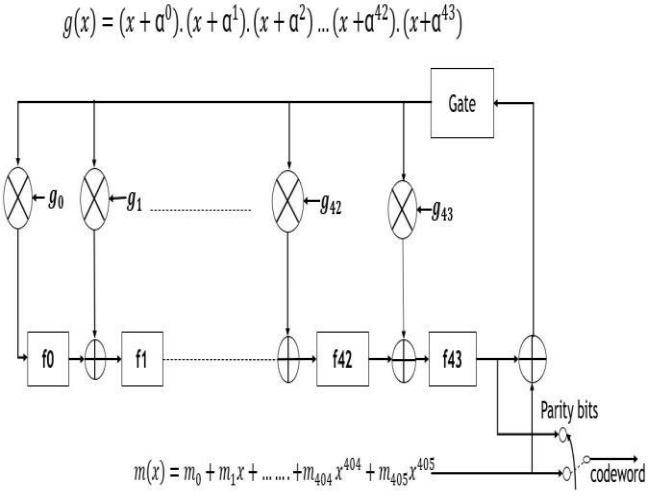


Fig. 4. Architecture Of RS(450, 406) encoder.

The Fig. 4 shows the architecture of the Reed Solomon encoder, where $g(x)$ is the generator polynomial of the shortened RS(450,406) encoder and $m(x)$ is the message symbols. This encoder circuit is a group of linear shift register[14], $f0, f1, f2, \ldots, f43$, Galois multiplier unit, adder(XOR), gate and switch. The coefficient of the generator polynomials are $g_0, g_1, g_2, \ldots, g_{42}, g_{43}$ and that are connected with the multiplier unit. Gate will be open initially, so all linear shift registers are initialized with zero.

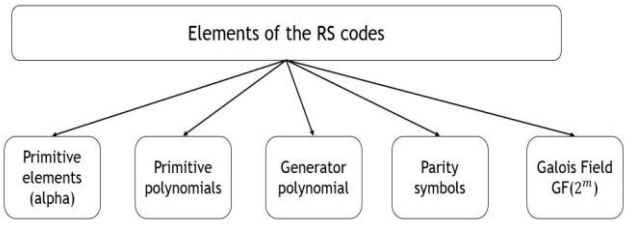


Fig. 5. Elements of RS codes.

Total n cycles are required to calculate the $n - k$ (44) parity symbols and these parity symbols are appended in original message.

For every positive clock cycle, message symbols are entered into the circuit and at the same time this message symbol are multiplied by coefficient of the generator polynomial. The output of the multiplier unit are stored in linear shift registers and output of the register $f43$ are added with the another message symbols, for another clock. This is the most important encoding techniques, in automotive[15] ethernet for forward error correction. The last 44 clock cycles are using for bringing the parity symbols from the shift registers. The main elements of Reed Solomon codes are shown in Fig. 5. Forward error correction techniques also used in satellite broad cast system[16]. The block diagram of second generation video satellite broad cast system is shown in Fig. 6.

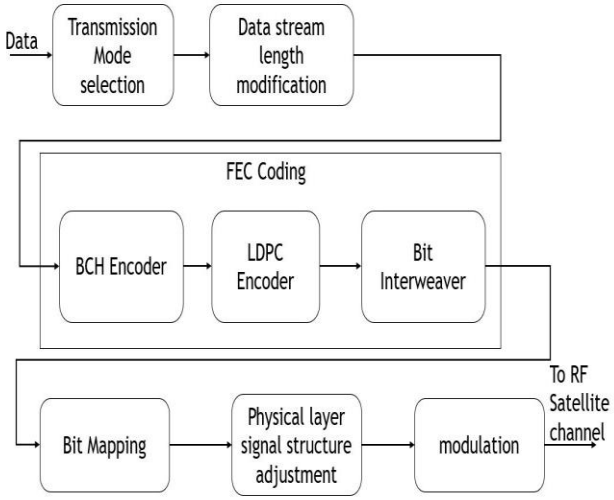


Fig. 6. Diagram of satellite broad cast system.

IV. IMPLEMENTATION & RESULTS

We implemented the Reed Solomon encoding in Verilog (Cadence ncsim) language and it is simulated in Cadence SimVision. We have also done its RTL (Register Transfer Level) coding and its RTL schematic is shown in Fig. 6.

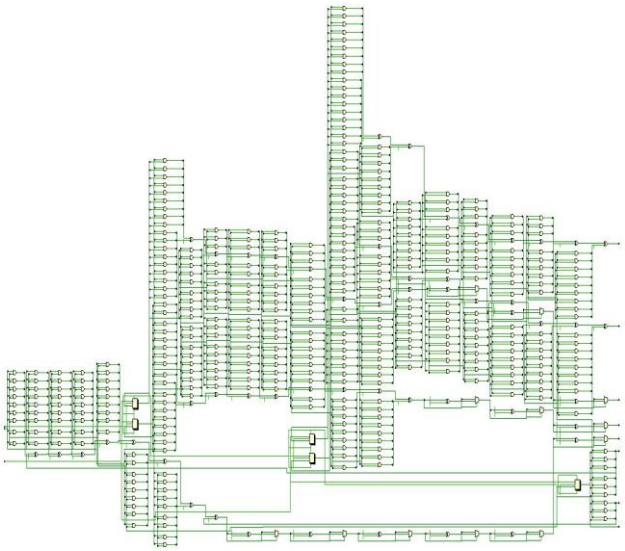


Fig. 6. RTL schematic of RS(450, 406) encoder.

The static verification has done through the Cadence linting tool HAL. The flow of the static verification is given in Fig. 7. If lint tool will found any design issues then modify the design otherwise sent it to next stage.

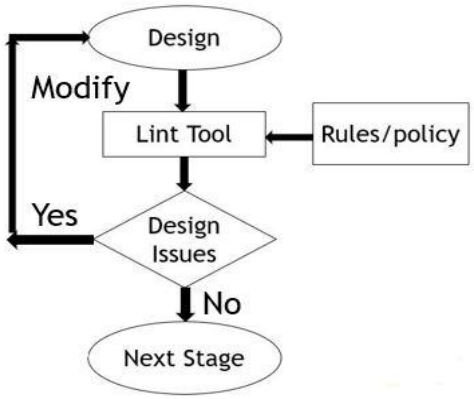


Fig. 7. Flow of static verification.

Total 406 message symbols (samp_data) are given to the encoder through the every positive clock as shown in Fig. 8.

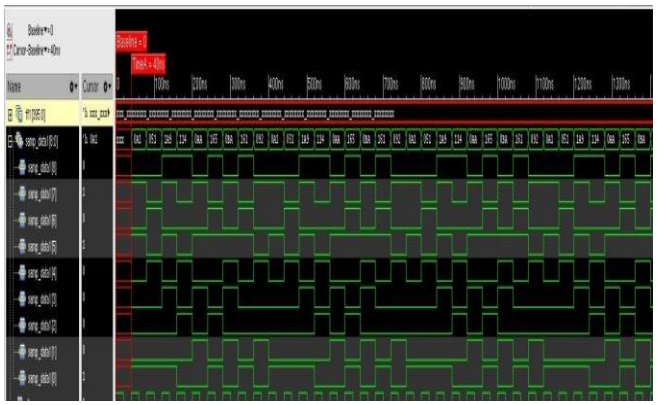


Fig. 8. Simulation waveform of message symbols.

The message symbols (samp_data) are 9-bits hexadecimal numbers started with $0A1_{HEX}$. Message symbols are multiplied by the coefficient of the generator polynomial and if the degree of the output multiplication polynomial is greater than or equal to the primitive then it will be divided by the primitive polynomial, its simulation is shown in Fig. 9. Where temp11 is the output of the multiplication and GF_multiplication is the output of the division.

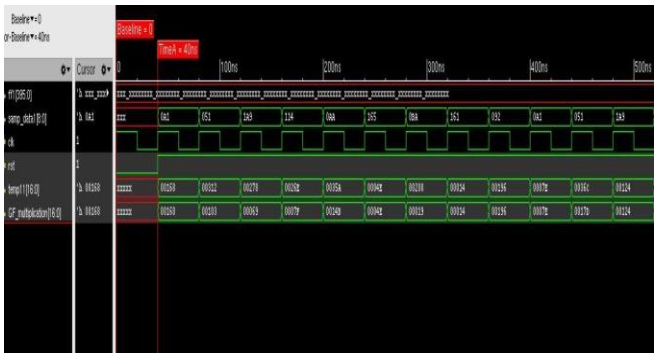
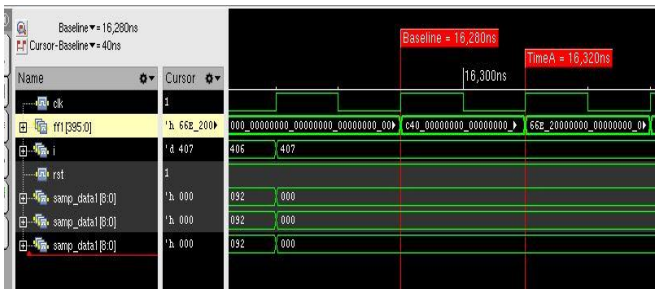


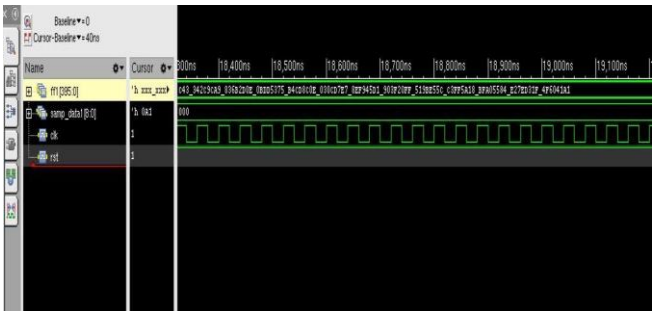
Fig. 9 simulation results of multiplication and division.

The reset signal (rst) is given to initialize the variables, and it is zero before the first positive edge of the clock and then it became one forever. Simulation results of the parity symbols is shown in Fig. 10.

After the k clock cycles (where k is equal to 406), $(n-k)$ clock cycles are required to calculate the parity symbols. So at the end of k cycle parity symbols are appending in one dimensional array $ff1$, where the length of $ff1$ is 396 as shown in Fig. 10(a). And after the n clock cycles (where n is 450), all parity symbols are appending in one dimensional array $ff1$ as shown in Fig. 10(b).



10(a)



10(b)

Fig. 10 Simulation results of parity symbols.

V. CONCLUSION

Reed Solomon code is the most powerful error correction and detection techniques used in 1000BASE- T1 to reduce the noise in communication channel. RS(450, 406) code is based on 9-bit Galois field $GF(2^9)$ and this RS code also called as shortened Reed Solomon code. In the vehicle network required more bandwidth due to more new technologies is used in the car and Gigabit automotive ethernet provides fast bandwidth in all kinds of advance driver systems. In this paper ,whole process of encoding is described and it is implemented in Verilog, simulated in Cadence SimVision. The coding efficiency of Reed Solomon code is increases with the code length. This Reed Solomon code is encoded on the basis of the generator polynomial and it can correct t random error symbols. The Reed Solomon code is better than other BCH code due to its efficiency and error correction capability. The Reed Solomon code is used in communication system, storage system and in Physical Coding Sublayer (PCS).

REFERENCES

1. Clarke, C. K., "Reed-Solomon Error Correction", British Broadcasting Corporation, July, 2002.
2. R. E. Blahut, Theory and Practice of Error-Control Codes. Reading, MA: Addison-Wesley, 1983.
3. M. A. Hasan, V. K. Bhargava, and T. Le-Ngoc, Reed- Solomon Codes and Their Applications, S. B. Wicker and V. K. Bhargava, Eds. Piscataway, NJ: IEEE Press, 1994. Algorithms and architectures for a VLSI Reed–Solomon code.
4. Kihoon Lee, Han-Gil Kang, et.al, "A High-Speed Low-Complexity Concatenated BCH Decoder Architecture for 100 Gb/s Optical Communications",. Springer Science, Business Media, August 2010.
5. Koetter R and Vardy A, "Algebraic soft-decision decoding of Reed–Solomon codes," IEEE Transaction on Information. Theory, vol. 49, no. 11, pp. 2809–2825, November 2003.
6. Lui W, Rho J, and Sung W, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in Proc. IEEE Workshop Signal Processing System, October 2006, pp. 303–308.
7. Sana Ullah and Mohammed A. Alnuem, "A Review of IEEE 802.15.6 MAC, PHY, and Security Specifications", Hindawi Publishing Corporation International Journal of Distributed Sensor Networks, article ID 950704, pp:1-12, 2013.
8. D. Gorenstein, N. Zierler, A Class Of Error-Correcting Codes In p^m Symbols, Journal of the Society for Industrial and Applied Mathematics, vol.9, no.2, pp.207-214, 1961.
9. B. Sklar, "Digital Communications: Fundamentals and Application," 2nd ed., Prentice Hall, 2001.
10. Omura, J., Massey, J.: Computational method and apparatus for finite field arithmetic. United States Patent 4587627 (May 6), 1986.
11. Onyszchuk, I., Mullin, R., Vanstone, S.: Computational method and apparatus for finite field multiplication. United States Patent 4745568 (May 2017), 1988.
12. 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.
13. Mustafa ELHAROUSSI, Asmaa HAMYANI, Mostafa BELKASMI ENSIAS RABAT MAROC, "VHDL Design and FPGA Implementation of a parallel Reed -Solomon (15,K,D) Encoder/Decoder" (IJACSA) 2013, Vol. 4.
14. D.Muthiah, A. Arockia Bazil Raj, "Implementation of High-Speed, LFSR Design with Parallel Architectures", International Conference on Computing Communication and Applications (ICCCA), pp. 1-6, Feb. 2012.
15. www.keysight.com/find/autonomous-driving.
16. Hagenauer, J., and Lutz, E., "Forward Error Correction Coding for Fading Compensation in Mobile Satellite Channels," IEEE JSAC, vol. SAC-5, no. 2, February 1987, pp. 215-225.

AUTHORS PROFILE

Mr. Akhilesh Yadav received the Bachelor of technology engineering in electronics engineering department from the Dr. A.P.J Abdul Kalam technical University Lucknow in 2108, Currently pursuing Master of technology in electronics and communication engineering department from the NIT Kurukshetra, and working as Student Intern in NXP Semiconductor Bangalore India. His research interest include Image processing, Computer Vision, Error correcting codes and Digital VLSI Design.

Dr. Poonam Jindal working since 2008 with ECE Department in National Institute of Technology Kurukshetra. She received her Ph.D. in Electronics and Communication Engineering from NIT Kurukshetra in 2016. She acquired degrees of M.Tech and B.Tech in 2005 and 2003 respectively. She has published 55 papers in various International Journals, Conferences and Book Chapters. Her research interests include wireless network security, wireless communication, physical layer security, Internet of Things, Security optimization in wireless networks. She is a reviewer of various reputed International Journals and conferences. She has guided 19 M.Tech. dissertations and 30 B.Tech. projects in the area of wireless networks.

Mr. Devaraju Basappa having an experience of 16 Years' in in ASIC front end design and implementation is currently working with NXP Semiconductor Bangalore as a Design Engineer. He received Bachelor of Engineering in Electronics and Communication.

Mr. Mahendra Prakashaiah having an experience of 19 Years' in VLSI Industry is currently working with NXP Semiconductor since 2018, as a Principal Engineer. He acquired degrees of M.Tech in Digital Electronics and Advanced communication from NIT Surathkal and Bachelor of Engineering in Electronics and communication from the UBDT College of Engineering Davanagere. During the period of Jan 2013 to Mar 2018 he worked in OMNIPHY INDIA PRIVATE LIMITED as an Architecture design and verification on USB2/USB3/PCIe/SATA/Ethernet/DDR PHY/Automotive Ethernet Phy, And during the period of Aug 2007 to Dec 2012 he worked in STMicroelectronics as Engineer Specialist with Lead HED/TVM IP verification Team.