International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 1, January- 2013

Implimentation Of Product Reed Solomon Code

K. Sailaja.
Asst.prof, ECE Dept
A.T.R.I
Hyderabad - 501401

N. Neelima
Associate Professor, ECE Dept
M.R.I.T.S
Hyderabad - 501401.

K.Y Srinivas
Associate.Prof, ECE Dept.
M.R.I.T.S
Hyderabad - 501401.

ABSTRACT

In this paper, we propose a (255, 247) product Reed-Solomon (RS) code for communication systems. Reed-Solomon codes are the most diversely used in communication systems, but powerful for burst errors only. In order to correct multiple random errors and burst errors, another efficient decoding algorithm are required. The product code composing of Reed-Solomon codes and shortened Reed-Solomon codes may allow decoding multiple errors beyond their error correction capability. The proposed code consists of two shortened Reed-Solomon codes and a conventional Reed- Solomon code. The proposed code can correct 4 symbol errors. Galois field arithmetic is used for encoding and decoding of Reed – Solomon codes. Galois field multipliers are used for encoding the information block. At the decoder, the syndrome of the received codeword is calculated using the generator polynomial to detect errors. Then to correct these errors, an error locator polynomial is calculated. From the error locator polynomial, the location of the error and its magnitude is obtained. Consequently a correct codeword is obtained.

I.INTRODUCTION

Digital communication system is used to transport an information bearing signal from the source to a user destination via a communication channel. The information signal is processed in a digital communication system to form discrete messages which makes the information more reliable for transmission. Channel coding is an important signal processing operation for the efficient transmission of digital information over the channel. It was introduced by Claude E. Shannon in 1948[1] by using the channel capacity as an important parameter for error – free transmission. In channel coding the number of symbols in the source encoded message is increased in a controlled manner in order to facilitate two basic objectives at the receiver: error detection and error correction A Reed-Solomon (RS) code is constructed in a Galois field. (GF(2^m)) [1][3] A RS code is a block code and can be specific as a cyclic (n, k) RS. The variable ‘n’ is the size of codeword by the symbol, ‘k’ is the number of data symbol and ‘2t’ is the number of parity symbols. Each symbol contains
‘m’ number of bits. The relationship between the size of symbol ‘m’ and the size of the codeword ‘n’ is given by n = 2^m-1. This means that if there are ‘m’ bits in one symbol, there could exist ‘2m-1’ distinct symbols in one codeword, excluding the one with all zeros. The RS code allows to correct up to t number of symbol errors where t is given by t = (n-k) /2..[4] The proposed code can correct errors by the burst error as well as by the multiple random errors. The proposed code may have much lower decoding complexity than that of a BCH code with the considerable decoder code rate. We employ three (255, 247) Reed-Solomon decoders to improve error rate against multiple random errors.

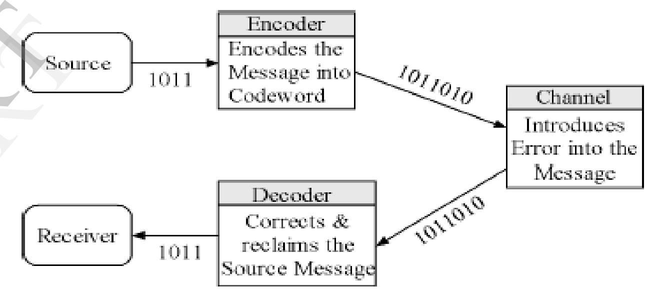
Source
1011
Encoder
Encodes the Message into Codeword
1011010
Channel
Introduces Error into the Message
1011010
Decoder
Corrects & reclaims the Source Message
1011
Receiver

Figure 1. Communication systems.

II.PROPOSED PRODUCT CODES

A. ***Encoder***

Transmitted data are systematically encoded. The codeword is represented as $c(X) = c_0 + c_1X + c_2X^2 + \ldots + c_{n-1}X^{n-1}$ (ci αGF (2m)). As previously mentioned, 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.[5] Therefore any RS encoder design should effectively perform the following two operations, namely division and shifting. Suggests that both operations can be easily implemented using Linear-Feedback Shift Registers.

The parity symbols are computed by performing a polynomial division using GF algebra. The steps involved in this computation are as follows:

Multiply the message symbols by X^{n-k}. This shifts the message symbols to the left to make space for the n-k parity symbols.

Divide the message polynomial by the code generator polynomial using GF algebra.

The parity symbols are the remainder of this division. These steps are accomplished in hardware using a shift register with feedback. The architecture for the encoder is shown

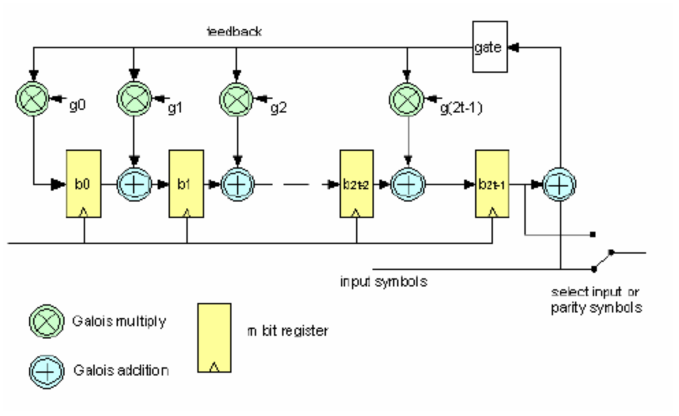

Fig:1 RS Encoder Circuitry

After first encoding, encoded data of containing extra 7 symbols are sent to a (255, 247) conventional RS encoder. The conventional RS encoder processes received data in different way compared to the shortened RS encoder. Therefore, before second encoding, the 247 data symbols must be rearranged with using the transfer block[6][7]. This transfer processing is different from first one. The transfer block code after the shortened RS encoder rearranges column wise to a row wise sequential bit block code.

The encoder consists of a duplicated structure using (255, 247) RS codes that have four error correcting capability. First part composes two (120, 118) shortened RS codes that encode input data is transmitted through shortened RS encoder processes 112 input data symbols from 1st to 112th, and remaining bits are transmitted by using#2 shortened RS encoder processes from 113th to 224th. The shortened RS encoder fills up 135 symbols with '0' symbols. Two shortened RS codes have a format of 255 byte codeword despite of containing 120 message symbols.

For this process, the input data transferred in row wise sequence are rearranged into a column wise vector before encoding into shortened RS codes as shown in Fig. 1. They restructure each message data for an encoder and decode in reverse sequence. For a shortened RS code, the transferred block rearranges the original data into a column-wise vector. For a (255, 247) RS code, it rearranges the column-wise data to the original data structure

When data transfer is ended, the conventional RS encoding is started. Because of using one RS coding processor, the conventional RS encoding process is the same as the shortened one ahead. The codeword is encoded against burst errors and multiple random errors. The parity symbols encoded by a conventional RS code have information about burst errors, especially. Furthermore, another parity symbols done by two shortened RS codes have information about multiple random errors, especially. The processing from preparing data to sending the codeword to decoder is shown in Fig. 2.

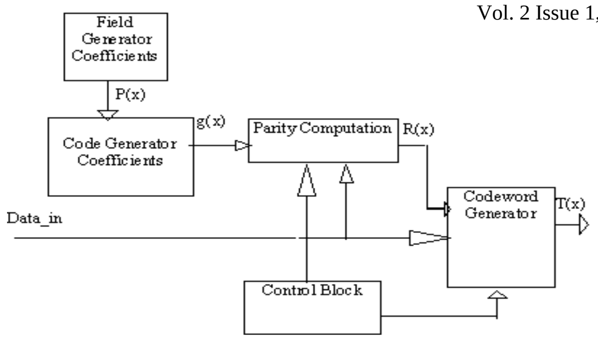

Fig: 2 Functional diagram of Encoder

B. Decoder

The decoding is processed in reverse order, a conventional (255, 247) RS decoding is processed in forward direction. It searches the data and corrects errors in row wise. At this point, the data structure is the same as the original, thus the decoder becomes to correct burst errors.

A typical decoder follows the following stages in the decoding cycle, namely

1. Syndrome Calculation
2. Determine error-location polynomial
3. Solving the error locator polynomial - Chien search
4. Calculating the error Magnitude
5. Error Correction

Figure 3 shows the internal flow diagram for the functional level depiction of the RS decoder unit. Rest of this section deals with various functional level stages shown in the diagram.

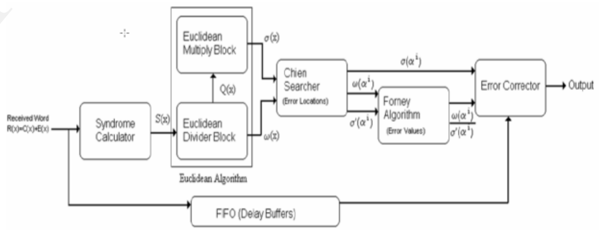

Fig:3 Block Diagram of a RS Decoder

1. SYNDROME CALCULATION

The *syndrome* is the result of a parity check performed on **r** to determine whether **r** is a valid member of the codeword set [3]. If in fact **r** is a member, the syndrome **S** has value **0**. Any nonzero value of **S** indicates the presence of errors. Similar to the**n** binary case, the syndrome **S** is made up of *n - k* symbols, {*Si*} (i = 1, …, *n - k*). Thus, for this (7, 3) R-S code, there are four symbols in every syndrome vector; their values can be computed from the received polynomial, **r**(*X*). Note how the computation is facilitated by the structure of the code, given by Equation (27) and rewritten below:[8][9]

$$\mathbf{U}(X) = \mathbf{m}(X)\,\mathbf{g}(X)$$

From this structure it can be seen that every valid codeword polynomial **U**(*X*) is a multiple of the generator polynomial **g**(*X*). Therefore, the roots of **g**(*X*) must also be the roots of **U**(*X*). Since **r**(*X*) = **U**(*X*) + **e**(*X*), then **r**(*X*) evaluated at each of the roots of **g**(*X*) should yield zero only when it is a valid codeword. Any errors will result in one or more of the computations yielding a nonzero result. The computation of a syndrome symbol can be described as follows:[8]

$$\mathbf{S_i} = r(X)|_{X=\alpha^i} = r(\alpha^i) \qquad i=1,2\ldots\ldots\ldots\ldots n-k$$

Where r(X) contains the postulated two symbol errors as shown in equation. If r(X) were a valid codeword, it would cause each syndrome symbol S_i to equal 0.

II.ERROR LATION POLYNOMIAL

Suppose there are ν errors in the codeword at location $X^{j1}, X^{j2}, \ldots, X^{jv}$. Then, the error polynomial **e**(*X*) can be written as[5][6]

$$e(X) = e_{j1}X^{j1}+e_{j2}X^{j2}+\ldots\ldots\ldots\ldots\ldots+e_{jv}X^{jv}$$

The indices 1, 2, … ν refer to the first, second, …, νth errors, and the index *j* refers to the error location. To correct the corrupted codeword, each error value e_{jl} and its location X^{jl}, where $l = 1, 2, \ldots, \nu$, must be determined. We define an error locator number as $\beta_l = \alpha^{jl}$. Next, we obtain the $n - k = 2t$ syndrome symbols by substituting α^i into the received polynomial for $i = 1, 2, \ldots 2t$:

$$S_1 = r(\alpha) = e_{j1}\beta_1+e_{j2}\beta_2\ldots\ldots\ldots\ldots+e_{jv}\beta_v$$

$$S_2 = r(\alpha^2) = e_{j1}\beta_1^{\,2}+e_{j2}\beta_2^{\,2}\ldots\ldots\ldots\ldots+e_{jv}\beta_v^{\,2}$$

$$S_{2t}= r(\alpha^{2t}) = e_{j1}\beta_1^{\,2t}+e_{j2}\beta_2^{\,2t}\ldots\ldots\ldots\ldots+e_{jv}\beta_v^{\,2t}$$

There are 2*t* unknowns (*t* error values and *t* locations), and 2*t* simultaneous equations. However, these 2*t* simultaneous equations cannot be solved in the usual way because they are nonlinear (as some of the unknowns have exponents). Any technique that solves this system of equations is known as a *Reed-Solomon decoding algorithm*.

Once a nonzero syndrome vector (one or more of its symbols are nonzero) has been computed, that signifies that an error has been received. Next, it is necessary to learn the location of the error or errors. An error-locator polynomial, **σ**(*X*), can be defined as follows:

$$\boldsymbol{\sigma}\mathbf{(X)} = (1+\beta_1X)(1+\beta_2X)\ldots\ldots\ldots\ldots\ldots(1+\beta_v)$$

$$= 1+\sigma_1X+\sigma_2X^2+\ldots\ldots\ldots\ldots\ldots\sigma_vX^v$$

The roots of **σ**(*X*) are 1/β1, 1/β2, … ,1/βν. The reciprocal of the roots of **σ**(*X*) are the error-location numbers of the error pattern **e**(*X*). Then, using autoregressive modeling techniques [7], we form a matrix from the syndromes, where the first *t* syndromes are used to predict the next syndrome. That is,

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_{t-1} & S_t \\ S_2 & S_3 & S_4 & \cdots & S_t & S_{t+1} \\ & & & \vdots & & \\ S_{t-1} & S_t & S_{t+1} & \cdots & S_{2t-3} & S_{2t-2} \\ S_t & S_{t+1} & S_{t+2} & \cdots & S_{2t-2} & S_{2t-1} \end{bmatrix} \begin{bmatrix} \sigma_t \\ \sigma_{t-1} \\ \vdots \\ \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{t+1} \\ -S_{t+2} \\ \vdots \\ -S_{2t-1} \\ -S_{2t} \end{bmatrix}$$

III. SOLVING THE ERROR LOCATOR POLYNOMIAL_CHIEN CEARCH

An error had been denoted e_{jl}, where the index *j* refers to the error location and the index *l* identifies the lth error. Since each error value is coupled to a particular location, the notation can be simplified by denoting e_{jl}, simply as e_1. Preparing to determine the error values e_1 and e_2 associated with locations $\beta_1 = \alpha^3$ and $\beta_2 = \alpha^4$,[7]

Any of the four syndrome equations can be used. Let's use *S*1 and *S*2.

$$\mathbf{S_1=r(\alpha)=}e_1\beta_1+e_2\beta_2$$

$$\mathbf{S_2=r(\alpha^2)=}e_1\beta_1^{\,2}+e_2\beta_2^{\,2}$$

IV.CALUCULATING ERROR MAGNITUDE

The estimated error polynomial is formed, to yield the following:

$$\hat{\mathbf{e}}(X) = e_1X^{j_1} + e_2X^{j_2} = \alpha^2X^3 + \alpha^5X^4$$

The demonstrated algorithm repairs the received polynomial, yielding an estimate of the transmitted codeword, and ultimately delivers a decoded message. That is,[10]

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

V. Correct the received word C (x) = E(x) + R (x)

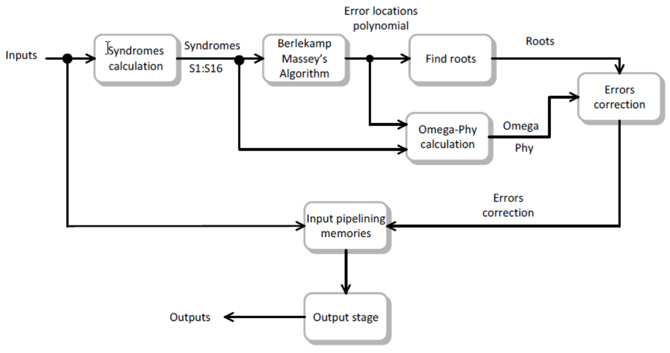

Fig: 4 Decoder

III.SIMULATION RESULT FOR REED SOLOMON CODES

The simulation results for RS encoder including input signal valid. If valid signal is low that means data input is not valid, data is encoded only when this signal goes high. Moreover, all the inputs and outputs go low when reset is de-asserted. Inputs g0 to g15 are outputs from code generator.

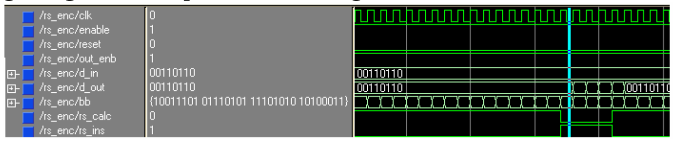
```
/rs_enc/clk        0
/rs_enc/enable     1
/rs_enc/reset      0
/rs_enc/out_enb    1
/rs_enc/d_in       00110110
/rs_enc/d_out      00110110
/rs_enc/bb         {10011101 01110101 11101010 10100011}
/rs_enc/rs_calc    0
/rs_enc/rs_ins     1
```

Figure5: Simulation Results of Entity Encoder

the simulation results for RS decoder including input signal start. Decoding will be started when start goes from high to low availing ready assertion. Decoding will be stopped immediately after the ready is de-asserted as shown in a fig

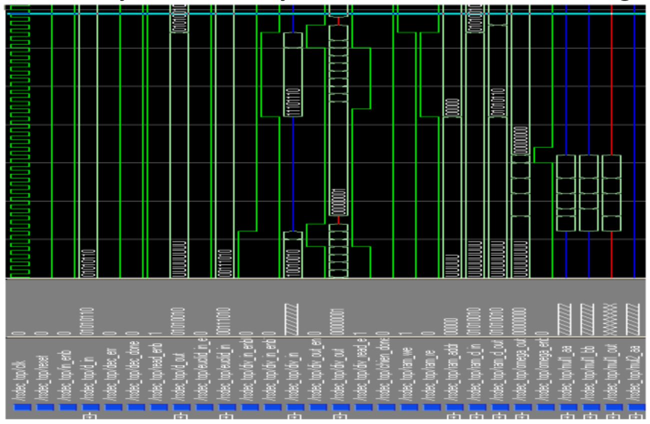

Figure- 6 Simulation Results of Decoder

IV. CONCLUSION

This paper proposes a (255, 247) product Reed-Solomon Code for multiple random errors and burst errors. The proposed code takes two dimensional array data consisted of two shortened Reed-Solomon codes in a column-wise and a conventional Reed-Solomon code in a row-wise. The proposed code becomes powerful against multiple random errors and burst errors. The proposed code can corrects 4 symbol errors.

REFERENCES

[I] E. Savas, A. F. Tenca and C.K. Koc. A Scalable and Unified Multiplier Architecture for Finite Fields GF(p)and GF(2m) [J]. In C.K. Koc and C. Paar (Eds.):Cryptography Hardware and Embedded Systems, CHES 2000, LNCS, Springer-Verlag, pp.277-292, 2000

[2] E.D. Mastrovito, VLSI Design for Multiplication over Finite Fields[J]. Springer-VerlagJuly 1988. pp. 297-309.

[3] E.D. Mastrovito. VLSI Architectures for Computations in GaloisFields[M]. Sweden: Linkoping Univ. Linkoping, 1991.

[4] S.TJ. Fenn, M. Benaissa, and D. Taylor. Improved Algorithm for Division over GF(2j)[M]. Electronic Letters.Mar. 4,1993. vol. 29, pp.469-470.

[5] Zhu Yehua,Ding Jie,Ke Jiandong,Liu Wenjiang, DMT processor RS decoder design [J],Telecommunication tecknology,2006 02

[6] Wang Jinxiang,Zhang Naitong,Ye Yizheng, RS (255,223) encoder design and CPLD implementation [J]; micro-electronics; 1999 05.

[7] Xu Feng; Li Jian; Vo Van Hung;; FPGA-based RS (255,223) encoder design [J]; micro-computer information; 2006 26.

[8] H. M. Shao and T. K. Truong, “A VLSI Design of a Pipeline Reed Solomon Decoders,” *IEEE Trans. Comput.*, pp. 393-403. Mar. 1985.

[9] D. V. Sarwate and N. R. Shanbhag, “Hign Speed Architectures for Reed-Solomon Decoders,” *IEEE Trans. On VLSI Systems*, vol. 9, No. 5, pp. 641-655, 2001.

[10] R. E. Blahut, *Theory and practice of error-control codes*, Reading, MA: Addison-Wesly, 1983.

[11] S. G. Wilson, *Digital Modulation and Coding*, Prentice Hall, 1996.

[12] Altera Corporation. Stratix data sheet, May 2003
.

[13] Altera Corporation. Nios stratix devleopment kit, July 2003.

[14] M. Mehnert, D. F. von Droste, and D. Schiel, “VHDL Implementation of a (255, 191) Reed Solomon Coder for DVB-H,” *IEEE 10th International Symposium on Consumer Electronics (ISCE)*, pp. 1-5, 2006.