# Design and Formulative Analysis of VLSI Syndrome Generator for RS(128,$K_x$) and RS(64,$K_y$)

Remalli Dinesh
Student member, IEEE
Lovely Professional University
Jalandhar, India

Sandeep Bansal
Department of E.C.E
Lovely Professional University
Jalandhar, India

## ABSTRACT
A VLSI architecture of Syndrome generator is part of well known Reed Solomon codes which is capable of generating syndromes while receiving RS (N, K) Codes from the transmitter. RS codes are customized to achieve a suitable VLSI realization of syndromes which can be further used in error detecting algorithms such as Euclidean, Erasures, Berlekamp's iterative algorithm and Massey Linear feedback shift register synthesis algorithm. It is assumed to be one of the pipelining stages in the Reed Solomon decoding technique. A MATLAB implementation of syndrome generator with various RS codes having different ranges of bit symbols and codeword was reveal for the purpose of determining syndromes. Hardware intricacy depends only on 2p parity check bytes. Message is encoded in Galois field having 2^m elements. Each Codeword enclose 'N' symbol generates syndromes for those 'N' symbols. It represented implementation of RS (128, Kx) and RS (64, Ky) with the different values of bits/symbol 'm' and different range of $K_x$ & $K_y$ and look into how limit of Zeros and Syndrome varies.

## Keywords
Syndromes, Reed Solomon decoders, pipelining processing, Galois field, VLSI, error detection, generator polynomial.

## 1. INTRODUCTION
Syndrome generator has been extensively used in an array of Reed Solomon decoder's circuit. In decades, from the time when the discovery of Reed Solomon codes have benefitted with numerous applications, from CD players to Spacecrafts. As it looks after digital data from counter to errors during transmission. Since the syndromes is determined by using syndrome polynomial which is degree of how far away the received code word is from the one that was transmitted [1]. The RS decoder utilizes concept of syndrome basically consist of three modules [6]. First module is a Syndrome calculator. This module generates the syndrome polynomial which can be shown below [1]

$$S(x) = R(x) \mid G(x) \mid \qquad (1)$$

The Polynomial generated is applied in second module for resolve the Key equation [2]. Euclidean and Berlekamp-Massey algorithm are consider for polynomial which inform us about the error location and error magnitude of the RS code received on the receiver. Then in third module, equivalent values of error location polynomial and error magnitude polynomial are computed using various other techniques.

In this paper, shown the implementation of Syndrome for error correcting Reed Solomon codes, RS(128,$K_x$) and RS(64,$K_y$) decoders which can be operated on several devices where security and significance of data is given prime meaning.

Both arbitrary and disintegrate form of codes include errors due to communication channel, syndromes polynomial for this is generated and fed to error correcting algorithms[7]. For consistent transmission of data, RS codes with significant error correcting capability need to apply [3]. RS decoders containing Syndrome generators have the possible major improvement of cutback in size, load and power utilization. Despite the fact that, VLSI implementation makes available higher consistency at the same time as compared to regular sized isolated logic circuits.

An (n, n-2p) RS code is classified over the finite Galois field GF(2^m), where each code block contain 2p parity symbols and n is length of code block. The generator uses this information such that 't' errors that are present during transmission, so it can be further identify and correct in error correcting blocks. The code block can be correctable only if it follows the defined expression.

$$n - 2p < n \qquad (2)$$

$$p = \mu t \qquad (3)$$

where,

$$0 < \mu \leq 1 \qquad (4)$$

and for the accurate error correcting capability, $\mu$ =1.

This paper follows the architecture of finite field Galois element, generator polynomial and the encoded message through RS encoder. It use these encoded messages to decoder architecture and in that syndrome generator is implemented generating syndrome polynomial.

## 2. CONSTRAINT USED
The Syndrome generator uses RS code can be express with the following constraints and notation [4]:

m → the number of bits/symbol which lies in the range $3 \leq m \leq 16$

n → the number of symbols/codeword which lies in range $3 \leq n \leq 2\,\hat{}\,(m-1)$

p → the number of error correctable capability symbol errors

w → the number of words used by the Reed Solomon Encoder to encode before transmission the data sequences.

2p → the number of parity check symbols in the transmitted codeword

k → the number of symbols/message in the transmitted codeword

C(x) → the illustration of code block of the range of N-1 polynomial

R(x) → the received polynomial received at the receiver

T(x) → the transmitted polynomial at the transmitter

G(x) → the generator polynomial

S(x) → the Syndrome Polynomial

# 3. SYNDROME CALCULATOR BLOCK

## 3.1 Procedure

The Galois field GF($2^m$) of the information message was calculated and it has $\alpha,\alpha^2,\ldots,\alpha^{2p}$ as its polynomial roots[8]. And, the code to be sent were created using the Reed Solomon encoder and the generator polynomial g(x) was given by [5]:

$$g(x)=(x-\alpha)(x-\alpha^2).....(x-\alpha^{2p}) \quad (5)$$

The transmitted efficient code vector is given by [5] :

$$T(x)=T_0+T_1x^1+T_2x^2+.....+T_{N-1}x^{N-1} \quad (6)$$

The error caused by due to various channel, environmental and transmission medium are represented by systematic vector equation [5]:

$$e(x)=e_0+e_1x+e_2x^2+.....+e_{N-1}x^{N-1} \quad (7)$$

The received code vector at the input of the receiver consisting of transmitted signal come together with the error signal is represented as [5]:

$$r(x)=T(x)+e(x) \quad (8)$$

And,

$$r(x)=r_0+r_1x+r_2x^2+.....+r_{N-1}x^{N-1} \quad (9)$$

After getting the received vector, it is required to find Syndrome's using the following equation [3]:

$$S_i=\sum_{j=0}^{N-1} r_j\alpha^{(k+i)j} \quad (10)$$

Finally, the Syndrome polynomial required to find out from the above obtained Syndrome, by using the following equation [3]:

$$S(x)=\sum_{k=0}^{k=2p-1} S_kx^k \quad (11)$$

Now, the achieved Syndrome polynomial is used in finding syndromes for RS(128,$K_x$) and RS(64,$K_y$).

## 3.2 Design Methodology of Syndrome flow

The basic flow for data modeling before sending the Reed Solomon encoded codes using transmitter were shown below in flow chart:

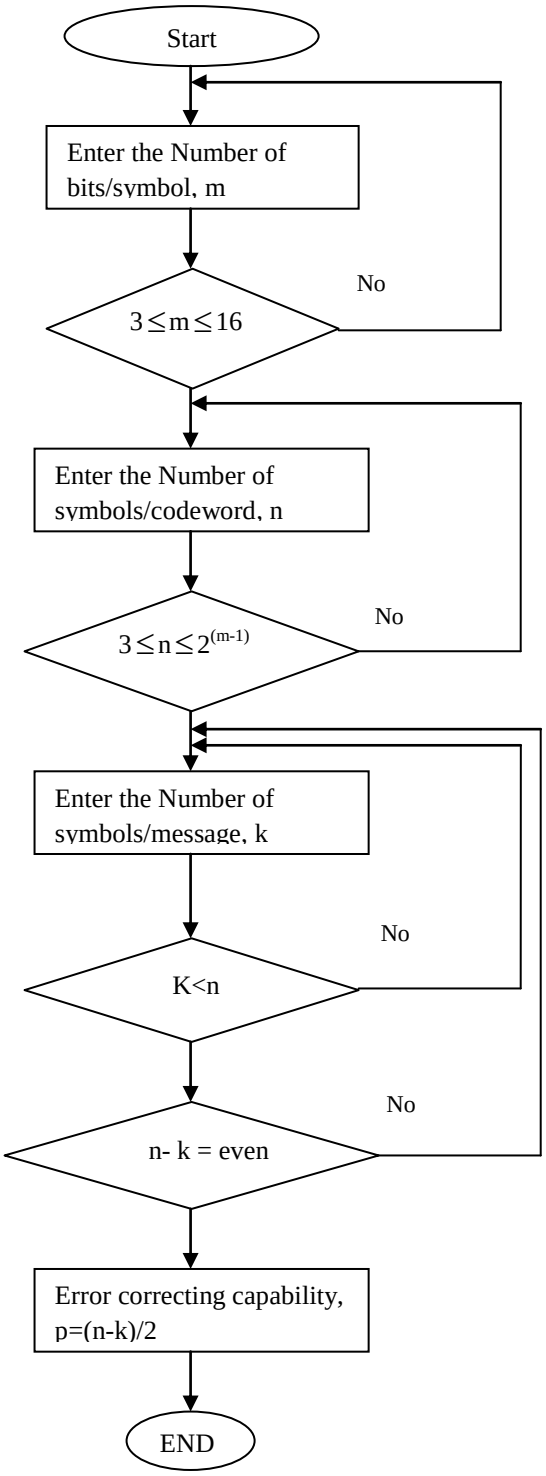


**Fig 1: Flow chart of Data flow in system**

The complete Reed Solomon codes with accurate error correcting capability factor and the syndromes generation using that encoded message are shown in the following block diagram:

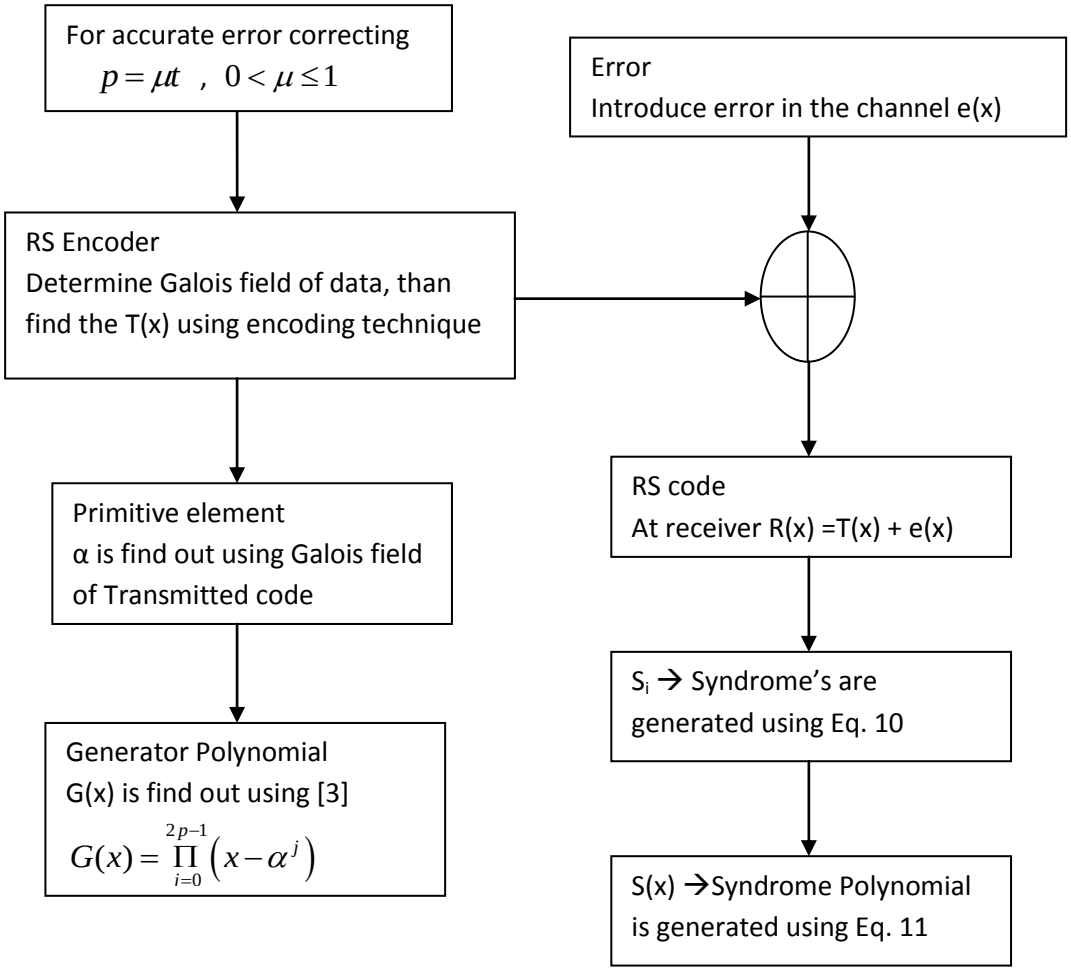


**Fig 2: Block diagram depicts flow from RS Encoder to Syndrome Polynomial Generation**

## 4. SIMULATION AND FINDINGS

In order to authenticate the performance of syndrome generator, a program was written with number of words, w=8 to replicate the system. The random generation of error pattern is introduced and the error positions are arbitrarily chosen by the computer. For fixed values of m=10, m=8 and different values of $K_x$ and $K_y$, we are getting different MAX values of $S_i$ in each case and lots of zeros. Here, zeros signify that the message is translated into syndrome without any sort of error.

### 4.1 Reed Solomon –RS(128,$K_x$)

**Table 1. Simulation values for varying $K_x$**

| $K_x$ | m = 10 | | m = 8 | |
|---|---|---|---|---|
| | $S_i$ = | Total number of zeros | $S_i$ = | Total number of zeros |
| $K_1$ = 8 | 93 | 498 | 105 | 520 |
| $K_2$ = 16 | 697 | 502 | 104 | 519 |
| $K_3$ = 32 | 454 | 509 | 243 | 531 |
| $K_4$ = 64 | 357 | 490 | 109 | 515 |
| $K_5$ = 96 | 609 | 514 | 241 | 526 |

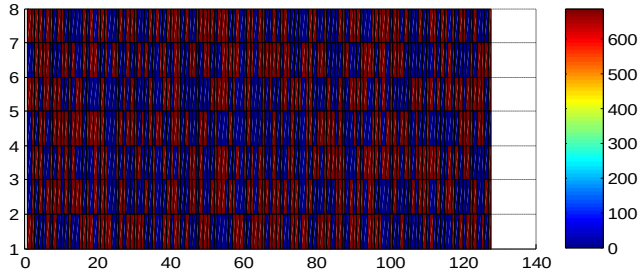


**Fig 3: Data flow in RS (128,8) using m=8**

### 4.2 Reed Solomon –RS (64, Ky)

**Table 2. Simulation values for varying $K_y$**

| $K_y$ | m = 10 | | m = 8 | |
|---|---|---|---|---|
| | $S_i$ = | Total number of zeros | $S_i$ = | Total number of zeros |
| $K_1$ = 4 | 829 | 268 | 185 | 261 |
| $K_2$ = 8 | 665 | 262 | 152 | 259 |
| $K_3$ = 16 | 674 | 263 | 173 | 257 |
| $K_4$ = 32 | 870 | 264 | 245 | 255 |
| $K_5$ = 48 | 765 | 273 | 178 | 262 |

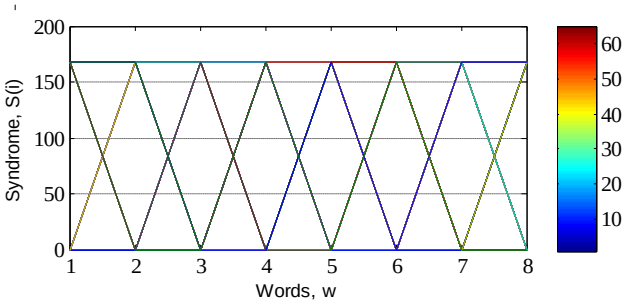


**Fig 4: Syndromes for RS (128,8) using m=8**

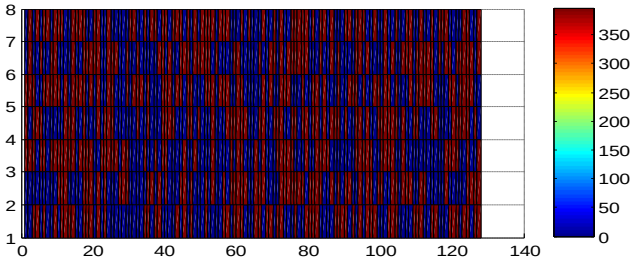

**Fig 5: Data flow in RS(128,8) using m=10**

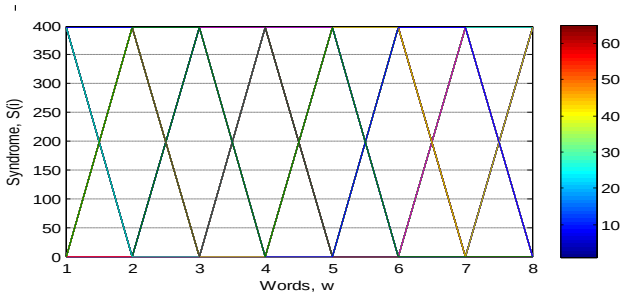


**Fig 6: Syndromes for RS(128,8) using m=10**

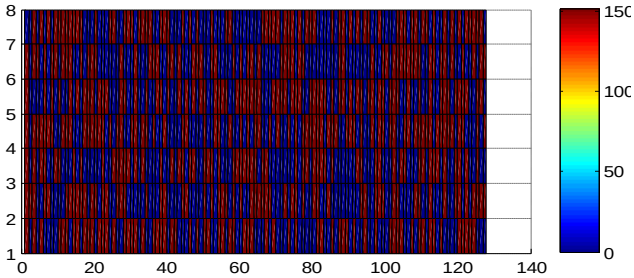

**Fig 7: Data flow in RS(128,16) using m=8**

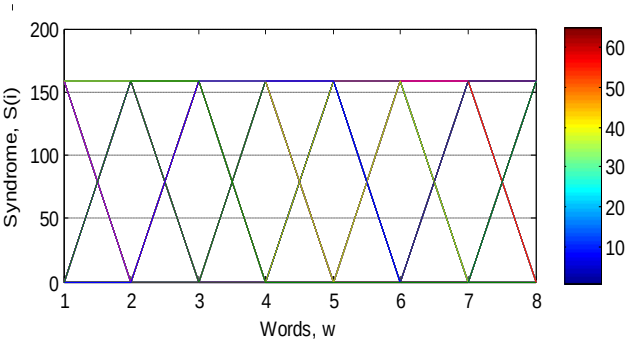


**Fig 8: Syndromes for RS(128,16) using m=8**

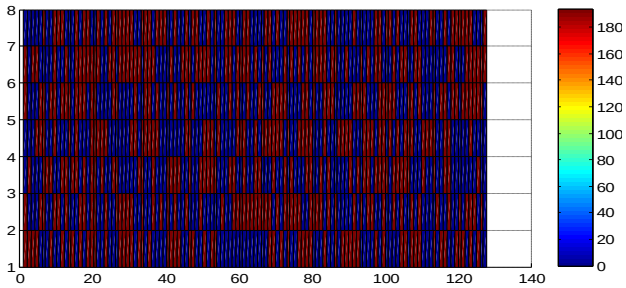

**Fig 9: Data flow in RS (128, 16) using m=10**

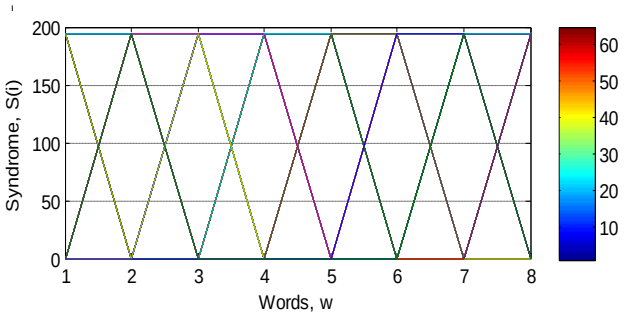


**Fig 10: Syndromes for RS (128, 16) using m=10**

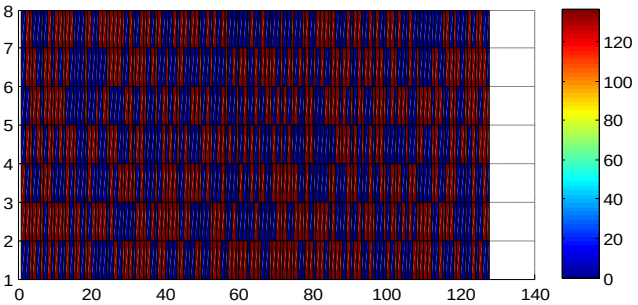

**Fig 11: Data flow in RS (128,32) using m=8**

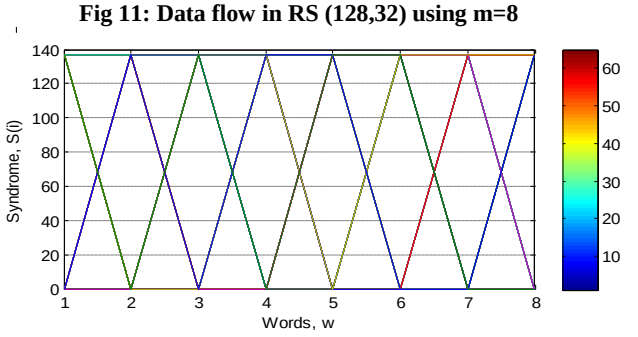


**Fig 12: Syndromes for RS(128,32) using m=8**

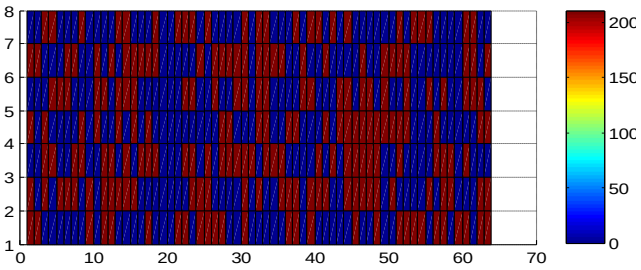

**Fig 13: Data flow in RS(64,4) using m=8**

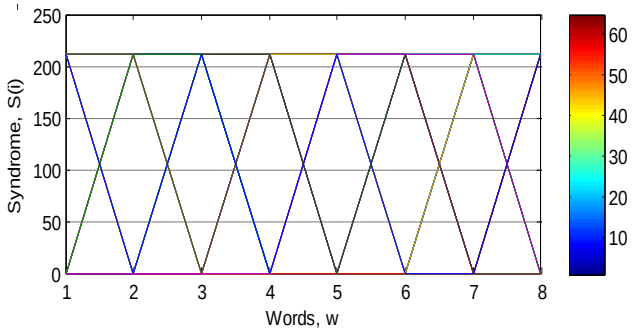


**Fig 14: Syndromes for RS(64,4) using m=8**

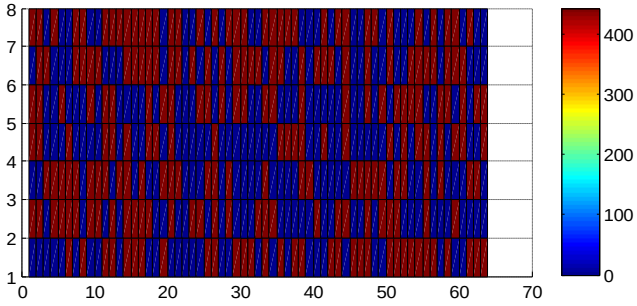

**Fig 15: Data flow in RS(64,48) using m=10**

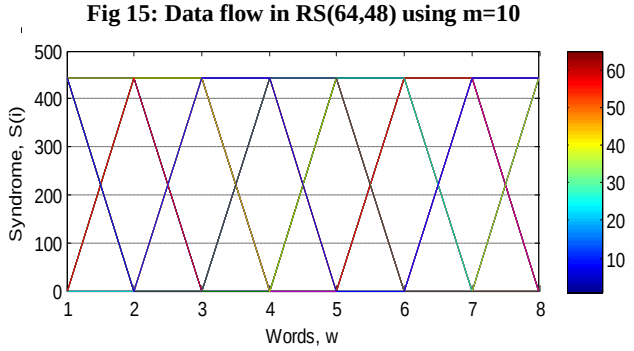


**Fig 16: Syndromes for RS(64,48) using m=10**

## 5. CONCLUSION

We have developed an implementation of VLSI Syndrome generator for RS(128,$K_x$) and RS(64,$K_y$). Pictures such as shown in Fig.3, Fig.5 and so on , it can observe that the x-axis represent the value of 'N' and it can be 64 or 128 and y-axis represents the number of words ( w=8) being transmitted from the transmitter. Also, the blue vertical bars correspond to number of zeros and the red vertical bar denotes the Syndrome, $S_i$ . From Table 1 and Table 2 , we conclude that if for twice the value of N, than for the same value of 'm' get almost double the number of zeros and after analyze RS(64,$K_y$) for the same value of 'N, but different value of m=10 and m=8, we get change in Syndrome almost thrice than with the increasing value of $K_y$. Also for RS(128,$K_x$), the number of zeros is more for low value of m and it increases with the increasing value of $K_x$. Finally, come up with a conclusion that the increase in the value of N and K in RS(N,K) , than the number of zeros increases with the increase in Syndrome, $S_i$ and it will be beneficial upto a certain limit .Because after that, more syndromes can cause outflow of information.

## 6. ACKNOWLEDGEMENTS

Our Precious gratitude intended for the fellow mentor and concerned people because without their help, it is very complicated to complete the work presented in this paper.

## 7. REFERENCES

[1] Gary K. Maki, Patrick A. Owsley, Kelly B. Cameron and Jack Venbrux, 1986, "VLSI REED SOLOMON DECODER DESIGN". University of Idaho, Moscow

[2] Hanho Lee, Meng-Lin yu and Leilei Song, "VLSI DESIGN OF REED -SOLOMON DECODER ARCHITECTURES". IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

[3] KUANG YUNG LIU, "Architecture for VLSI Design of Reed – Solomon Decoders". IEEE Transactions on Computers, VOl. C-33, Feb.1984.

[4] Hung-Wei Chen, Jiin-Chuan Wu, Gwo-Sheng Huang, Ji-Chien Lee and Shin-Shi Chang, IEEE 1995, "A New Vlsi Architecture of Reed-Solomon Decoder with Erasure Function".

[5] Ching-Lung Chi and Tsung-Hsiu Chi, 2011 "A Fast Reed-Solomon Decoder using Step-by-Step Algorithm".

[6] You yu-xin, WANG Jin-Xiang , LAI Feng-Chang and YE yi-zheng, 2002 " VLSI Design and Implementation of high speed RS(204,188) Decoder".

[7] Richard Huynh, GE Ning and YANG HuaZhong, "A Low Power Error Detection in the Syndrome Calculator Block for Reed-Solomon Codes: RS(204,188)" Vol 14, August 2009, Tsinghua Science and Technology.

[8] Dong-Sun Kim, Jong-Chan Choi and Duck-Jin Chung, IEEE 1999, "Implementation of High Speed Reed-Solomon Decoder".