IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735. Volume 6, Issue 5 (Jul. - Aug. 2013), PP 31-40
www.iosrjournals.org

Implementation of Product Reed Solomon Codes for Multi level cell Flash controller

[1]A.Vijaya Lakshmi
[1] *Associate Prof. Dept.ECE,VCE , JNTU, Hyderabad-AP,*

ABSTRACT: *In recent years, multi-level cell (MLC) flash memories have been developed as an effective solution for increasing the storage density and reducing the cost of flash memories. MLC flash memories are especially for NAND flash memories . Error control coding (ECC) is essential for correcting errors in Flash memories.In order to correct multiple random errors and burst errors, an efficient decoding algorithms are required.In this work (255, 231) Product Reed-Solomon (RS) code technique for non-volatile NAND flash memory system sare used. In this proposed code that is Product Reed Solomon can correct both multiple random errors and burst errors. The Product Reed Solomon code consists of two shortened Reed-Solomon codes and a conventional Reed- Solomon code. It can correct up to certain symbol errors. A simulation result shows that the code has improved the coding gain and low power consumption.*
Keywords: *Mlc Nand Flash Memory, Reed-Solomon Code*

Submitted Date 31 May 2013 **Accepted Date: 06 June 2013**

I. Introduction

Recently there has been an increasing demand for efficient and reliable data transmission and storage systems. This demand has been accelerated by the emergences of large scale, high speed data networks for exchanges, processing, and storage of digital information in the military, government and private spheres. A merging of communications and computer technology is required in the design of these systems. Recently, the demand for storage devices using NAND flash memory has been ever growing due to its attractive features such as low power-consumption, compactness, and high data throughput.[1,2] In spite of the growing popularity, expensive bit cost as compared to its competitors, e.g. magnetic storages, hinders storage devices using NAND flash memories from dominating the storage market. Motivated by that, semiconductor industry has been aggressively pursuing bit cost reduction, on one hand, through continuous technology scaling. NAND flash memory chips at 20nm technology are already widely used in industry. On the other hand, multi-level per cell (MLC) technology enabling a memory cell to store more than 1 bit has been developed to increase the storage density of NAND flash memory. In current design practice, 2bits per cell NAND flash memories are most prevailing in the market but possibilities of 3 and 4 bits per cell NAND flash memories are extensively explored [11], [17]. Data integrity is one of key requirements that storage devices must satisfy and generally ensured by relying on error-correcting codes (ECCs). However, design of ECCs for MLC NAND flash memory usually encounters the following technical challenges: 1) the target word-error rate/page-error rate (WER/PER) is usually set at an extremely low value like 10^{-14} or lower, 2) the length of message bits is rather long, typically 4K bytes or 8K bytes, 3) code rates are set relatively high, and 4) only hard-decision results are available when the error-control systems are off chip. For the case of single-level cells (SLC), 3- or 4-bit-error-correcting Bose-Chaudhuri-Hocquenghem (BCH) codes with a hard-decision decoding algorithm are enough to meet a required performance [1], e.g. a target PER. However, as the storage density gets higher, the reliability of the information bits stored in MLC NAND flash memory becomes gradually degraded and in turn deteriorates the data integrity.

In particular, the MLC technology and the technology scaling make the noise margin smaller and the cell-to-cell interference from adjacent cells stronger, respectively. To address these issues, as the storage density grows, ECCs have been consistently evolved to the ones with stronger error-correcting capabilities, e.g. 24- and 55-bit-error-correcting BCH codes for NAND flash memories are commercially available [16]. In addition, there have been efforts to design longer BCH codes for improving error correcting capability. Recently, BCH codes of lengths 1K [21] bytes and 4K bytes [3] are considered as error-correcting codes for MLC NAND flash memory. However, it is expected that the error-control systems using BCH codes soon will face a technological barrier since the decoding complexity of BCH codes quadratically grows with increasing error-correcting capability. The complexity issue becomes even worse at longer block lengths.[3,4]

Non-volatile NAND flash memory systems are widely used in the mobile and wireless systems. The requirement of high density and low cost makes the operation speed increased and simultaneously creates

various types of errors. With device scaling, each NAND flash cells are vulnerable to physical errors by coupling noises. As the memory density increases, its durability is weakened. These cause gate oxide degration that the bonding is easily broken. The multi-leveling cell (MLC), even though supplying powerful solutions, increases the performance of memory storage systems and causes many errors. In MLC, multiple bits are stored per a memory cell by each programming cell with multiple threshold levels. The errors are often generated by shift of threshold voltages during the operation. Therefore, the data reliability has become an important issue in most communication and storage systems for high speed operation and mass data process. Various error correction codes are provided for data reliability.

1.2. Our approach:

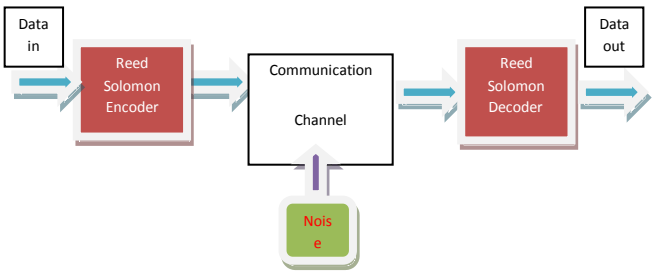
In this work, aiming at achieving a stronger error-correcting capability with much reduced complexity, we propose a high rate error-control system using The Product Reed Solomon code consists of two shortened Reed-Solomon codes and a conventional Reed- Solomon code. It can correct up to certain symbol errors.
The proposed code is Product Reed Solomon code scheme used for flash memories. This code can correct burst errors and multiple random errors. It may have much lower decoding complexity than that of a BCH code with the considerable decoder code rate. Here Product Reed Solomon codes are used to improve error rate against multiple random errors. A Reed-Solomon code is quite suitable for burst errors, but in case of random errors, it has some difficulty. BCH codes provide flexible code length and variable range of error correcting capability

II. Back Ground

BCH (Bose-Chaudhuri Hocquenghem) codes form a large class of powerful random error correcting cyclic codes .This class of codes is a remarkable generalization of the hamming codes for multiple error correction. Binary BCH codes were discovered by Hocquenghem and generalization of the binary bch codes to codes in P^m symbols (where p is a prime). BCH codes are a family of binary cyclic codes that are very flexible in terms of allowing a trade-off to be made between dimension and minimum distance. Also, for code lengths up to a few thousand, these codes are very efficient in terms of the performance trade-off that they offer.

2.1. Reed solomon codes

In 1960, Irving Reed and Gus Solomon published a paper in the Journal of the Society for Industrial and Applied Mathematics [1]. This paper described a new class of error-correcting codes that are now called Reed-Solomon (R-S) codes. These codes have great power and utility, and are today found in many applications from compact disc players to deep-space applications. This article is an attempt to describe the paramount features of R-S codes and the fundamentals of how they work.With this background in finite fields, we are now in a position to study the Reed-Solomon (RS) code. RS codes are in widespread use, for example in compact disks (CDs) and deep-space communication In contrast to the codes discussed thus far, these codes are non-binary. These codes have great power and utility, and are today found in many applications from compact disc players to deep-space applications. The RS decoder processes each block and attempts to correct errors and recover the original data as shown in fig 2.1.



2.1. A typical system using RS encoder and RS decoder

A Reed-Solomon code is a block code, meaning that the message to be transmitted is divided up into separate blocks of data as shown in fig 2.2. Each block then has parity protection information added to it to form a self-contained code word. A Reed-Solomon code is a systematic code.[5,6]

It is also, as generally used, a systematic code, which means that the encoding process does not alter the message symbols and the protection symbols are added as a separate part of the block.

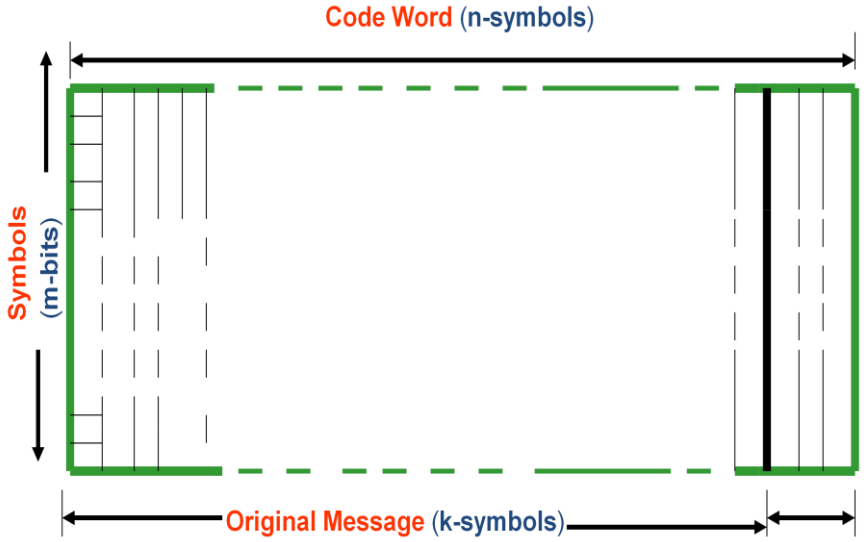


Fig.2.2 : Reed-Solomon code word

A Reed-Solomon code is a linear code It means that adding two code words produces another code word. A Reed-Solomon code is a cyclic code i.e cyclically shifting the symbols of a code word produces another code word. Reed-Solomon code belongs to the family of Bose-Chaudhuri-Hocquenghem (BCH) codes [3, 4], but is distinguished by having multi-bit symbols. This makes the code particularly good at dealing with bursts of errors because, although a symbol may have all its bits in error, this counts as only one symbol error in terms of the correction capacity of the code. Choosing different parameters for a code provides different levels of protection and affects the complexity of implementation.

2.2. Flash memory background

All computer-based systems contain memory. Memory is where information is stored while waiting to be operated on by the Central Processing Unit (CPU) of the computer. There are two types of memory. They are volatile memory and non-volatile memory. Volatile memory retains its information only while power is applied to the memory device. The contents of this memory type may be easily and quickly changed. Non-volatile memory retains its information even when no power is applied to the memory device. Although the information in most non-volatile memories may be changed, the process involved is much slower than for volatile memory.[7,8,9]

Non-volatile memory is memory that retains its contents even if the power is lost. Non-volatile memory was originally called Read Only Memory (ROM) because its contents were loaded during the manufacturing process and could be read, but never erased or reprogrammed. Over time, the ability to erase and reprogram ROM was added in different ways and referred to as Electrically Programmable ROM (EPROM), Electrically Erasable and Programmable ROM (EEPROM), and Flash EEPROM - commonly referred to simply as Flash memory. ROM memory is programmed by the way it is manufactured and stores permanent code and data that is generally used to initialize and operate a computer system.Flash memory has become the dominant type of non-volatile memory in use.

Flash memory is a non-volatile memory, which allows the user to electrically program (write) and erase information. The exponential growth of flash memory has made this technology an indispensable part of billions of electronic devices. Flash memory has several significant differences with volatile (RAM) memory and hard drive technologies which requires unique software drivers and file systems.

In addition to partitions, flash devices are further divided into sections of memory called blocks. Flash memory devices are available in symmetrical and asymmetrical blocking architectures . Devices with all blocks the same size are called symmetrically-blocked. Devices that are asymmetrically-blocked typically have several blocks that are significantly smaller than the main array of flash blocks. Small blocks or parameter blocks are typically used for storing small data or boot code. Block sizes vary but typically range from 64Kb to 256Kb.

This technical note discusses the basics of NAND Flash and demonstrates its power, density, and cost advantages for embedded systems. It covers data reliability and methods for overcoming common interface design challenges, focusing on the actual hardware and software components necessary to enable designers to build complete and functional subsystems. Embedded systems have traditionally utilized NOR Flash for non-volatile memory. Many current designs are moving to NAND Flash to take advantage of its higher density and lower cost for high-performance applications. shows how demand for NAND Flash has been driven primarily

several major markets,solid state drives, mobile phones, Flash memory cards, USB Flash drives and MP3/PMP players.[10,11]
As the quest has continued for lower-power, lighter, more robust products, NAND Flash has become the leading storage choice for a broad range of applications.

The NAND Flash device discussed in this technical note is based on a 2Gb asynchronous SLC device and its parameters (unless otherwise noted). Higher density devices and other more advanced NAND devices may have additional features and different parameters. The NAND Flash array is grouped into a series of blocks, which are the smallest erasable entities in a NAND Flash device .Nand flash architecture is as shown in fig.2.3
A NAND Flash block is 128KB. Erasing a block sets all bits to 1 (and all bytes to FFh). Programming is necessary to change erased bits from 1 to 0. The smallest entity that can
be programmed is a byte. Some NOR Flash memory can perform READ-While-WRITE operations. Although NAND FLASH cannot perform READs and WRITEs simultaneously,

It is possible to accomplish READ/WRITE operations at the system level using a method called shadowing. Shadowing has been used on personal computers for many years to load the BIOS from the slower ROM into the higher-speed RAM.

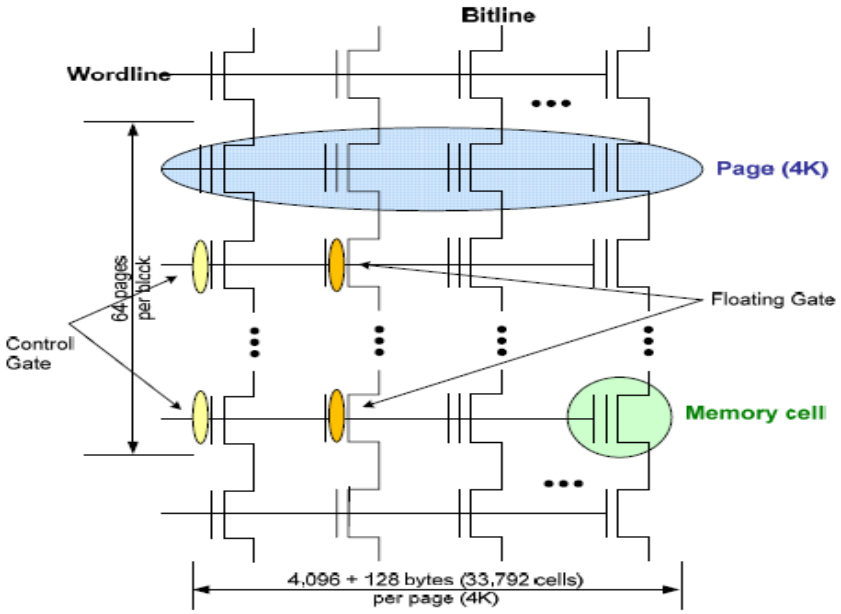


Fig.2.3. NAND Flash Architecture

There is a limit to the number of times NAND Flash blocks can reliably be programmed and erased. Nominally, each NAND block will survive 100,000 PROGRAM/ERASE cycles. A technique known as wear leveling ensures that all physical blocks are exercised uniformly. To maximize the life span of a design, it is critical to implement both wear leveling and bad-block management.a comparison of NAND Flash and NOR Flash cells. NAND efficiencies are due in part to the small number of metal contacts in the NAND Flash string. NAND Flash cell size is much smaller than NOR
Flash cells require a separate metal contact for each cell.

NAND flash is a high-density, non-volatile memory that requires increased algorithm complexity during factory programming when compared to other flash memory architectures, such as NOR. This is primarily due to the existence of bad memory blocks and other reliability issues in the device shipped from the supplier. NAND flash manufacturers are able to achieve commercially viable yields by allowing a small portion of the memory to fail test while still classifying the device as good. This tradeoff produces the very low-cost, high-density characteristics for which NAND flash is sought. By comparison, NOR flash is shipped with no such bad blocks at a much higher cost per bit and is simply not available in the densities that NAND provides. NAND flash is also superior to NOR for erase and write performance. By contrast, NOR is the fastest when it comes to reading the memory.

III. System Model

3.1. Constructing a product reed-solomon code

A Reed-Solomon (RS) code is constructed in a Galois field. (GF(2m). A RS code is a block code and can be specified as a cyclic (n, k) RS. The variable ‘n’ is the size of codeword by the ’k ‘ is the number of data symbol and the number of parity symbols is’2t’. 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=2m-1 That is ‘ m’ bits in one

symbol, there could exist , 2m-1 distinct symbols in one codeword. The RS code allows correcting up to t number of symbol Errors where t is given by t = (n-k) / 2. The codeword is represented as
$c(X) = c0 + c1x + c2x^2 + \ldots + cn\text{-}1X^{n-1}$ (ci € GF(2m)).

$c(X) = c0 + c1X + c2X^2 + \ldots + cn\text{-}1X^{n-1}$(ci) GF(2m)).

g(X) is generator polynomial that represents as below.
$g(X) = (X - \alpha)(X - \alpha 2)(X - \alpha 3)\ldots(X - \alpha 2t) = g0 + g1X + g2X2 + \ldots + g2t\text{-}1X2t\text{-}1 + X2t$ (gi α GF(2m)).

The Product Reed Solomon code block diagram is as shown in fig 3.1. The values of the message and parity symbols of a Reed-Solomon code are elements of a Galois field. Thus for a code based on m-bit symbols, the Galois field has 2m elements. The encoder consists of a duplicated structure using (255,247) RS codes that have four error correcting capability . First part compose two (120,112) shortened RS codes that encode input data in column wise.[16]
First shortened RS encoder processes 112 input data symbols from 1 to 112, and second shortened RS encoder processes from 113 to 224.the shortened RS encoder fills up to 135 with 0 symbols. Two shortened RS codes have a format of 255 byte code word despite of containing 120 message symbols .

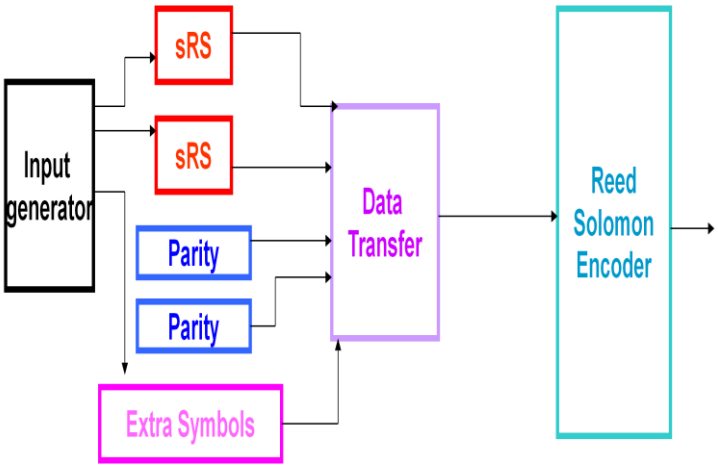


***Fig.3.1.*The block diagram of Product Reed-Solomon encoder.**

First shortened RS encoder processes 112 input data symbols from 1 to 112, and second shortened RS encoder processes from 113 to 224.the shortened RS encoder fills up to 135 with 0 symbols. Two shortened RS codes have a format of 255 byte code word despite of containing 120 message symbols .

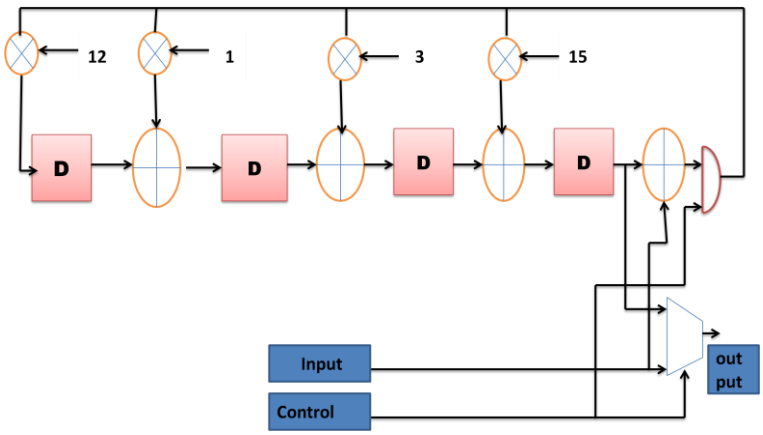


***Fig.3.2.*Internal encoding process**

X
The order of a (255, 239, t=8) product Reed-Solomon code, shortened to form a (204, 188, t=8) code, so that the 188 bytes of the input packet will be extended with 16 parity bytes to produce a coded block length of 204 symbols. For this code, the Galois field has 256 elements (m=8) and the polynomial representation of a field element is:

$a_7X^7 + a_6X^6 + a_5X^5 + a_4X^4 + a_3X^3 + a_2X^2 + a\ X^1 + a_0X^0$

Corresponding to the binary number 00000000 to 11111111.Alternatively, we can use the decimal equivalent 0 to 255. The specification also mention the field generator polynomial, given as

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

This allows us to construct a table of field element values for GF (256).[13] This shows how the field element values can be built up row by row for the 256 element Galois field in a similar manner to the construction of Table 1. At each step the binary number representing the 9 polynomial coefficient values is shifted to the left by one position (equivalent to multiplying by x) and 0 is added in the x0 column. If the shift causes a 1 to be lost at the left-hand side, then 00011101 is added to the columns, this being the substitution for X^8 obtained from the field generator polynomial (8) ,as it Clearly the full table for this field would be expensive.

shortened to form a (204, 188, t=8) code, so that the 188 bytes of the input packet will be extended with 16 parity bytes to produce a coded block length of 204 symbols. product Reed-Solomon code, shortened to form a (204, 188, t=8) code, so that the 188 bytes of the input packet will be extended with 16 parity bytes to produce a coded block length of 204 symbols. This polynomial is then divided by the code generator polynomial equation , to produce the four parity symbols as a remainder.

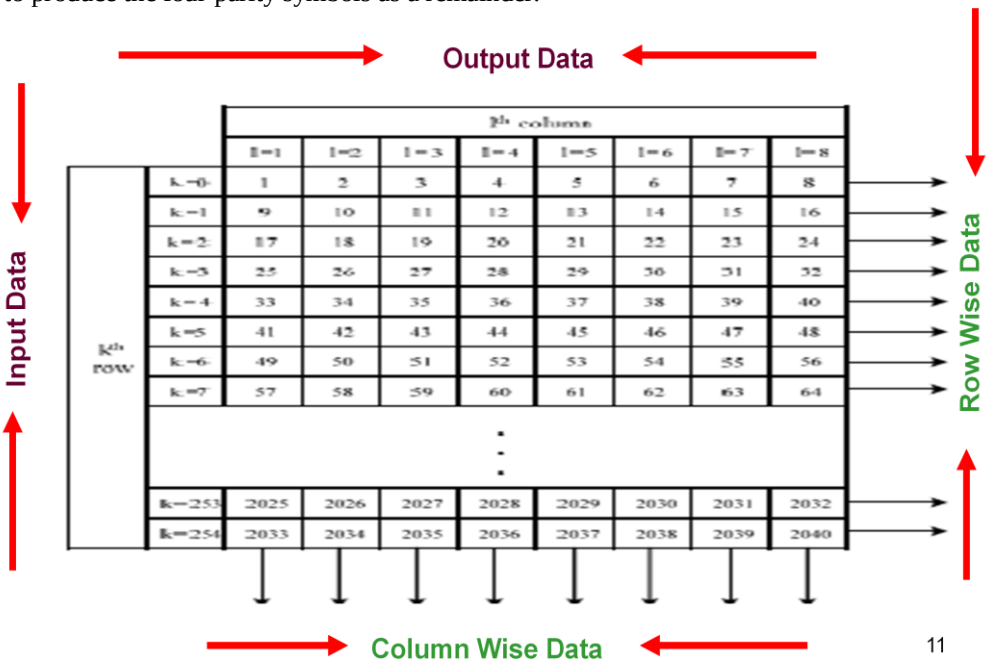


***Fig. 3.3.*Transferring row to column data**

3.2 Product Reed-Solomon decoding techniques:

Whereas the previous section has dealt with the underlying theory and, in some cases, identified several alternative approaches to some processes, this section will describe a specific approach to decoding hardware based around the Euclidean algorithm.

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 same as original, thus the decoder becomes to correct burst errors. The decoder first calculates the syndrome equations which shows any potential errors. if the syndrome polynomial s not zero , the receiving code word is erroneous then , the decoder corrects errors by calculating the error locator polynomial and the error evaluator polynomial . The process of decoding is to calculate error locator polynomial from the syndrome.

Adding the remainder, r(x), ensures that the encoded message polynomial will always be divisible by the generator polynomi al without remainder. [15]

$$M(X) \times X^{(n-k)} = g(X) \times q(X) + r(X)$$

and rearranging:

$$M(X) \times X^{(n-k)} + r(X) = g(X) \times q(X)$$

we note that the left-hand side is the transmitted code word, T(x), and that the right-hand side has g(x) as a factor. Also, because the generator polynomial, equation (6), ha s been chosen to consist of a number of factors, each of these is also a factor of the encoded message polynomial and will divide it without remainder. Thus, if this is not true for the received message, it is clear that one or more errors has occurred

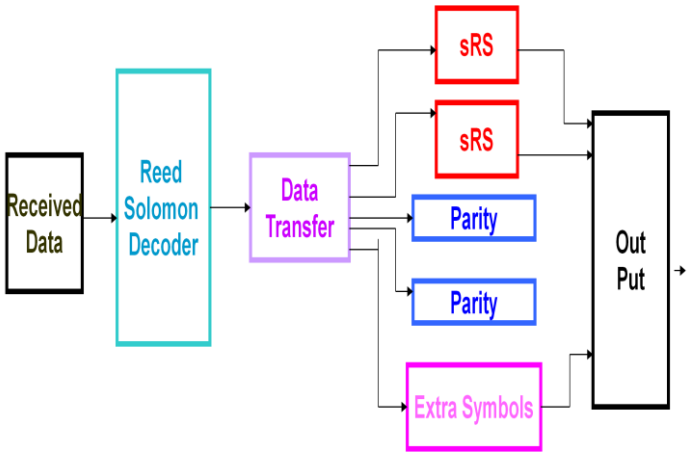


Fig.3.4 The block diagram of Product Reed-Solomon decoder

3.3. Main units of a Product Reed-Solomon decoder:

The arrangement of the main units of a Reed-Solomon decoder reflects, for the most part, the processes of the previous Section.

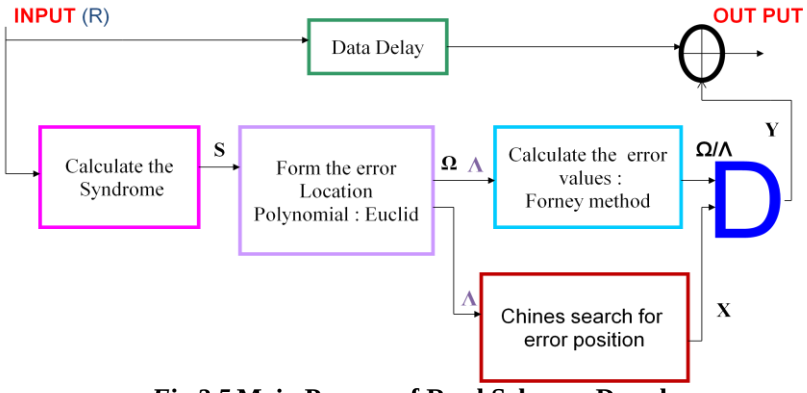


***Fig.3.5* Main Process of Reed Solomon Decoder**

IV. Results

Which contain real data captured on modelsim, it is combines high performance and high capacity with the most advanced code coverage and debugging capabilities in the industry. ModelSim offers unmatched flexibility by supporting 32 and 64 bit UNIX and Linux.As shown in fig the simulation result, Shows the input data sequence in the order of (255,231) which is given to the encoder input

PART-1: Encoder

4.1. Input Data:

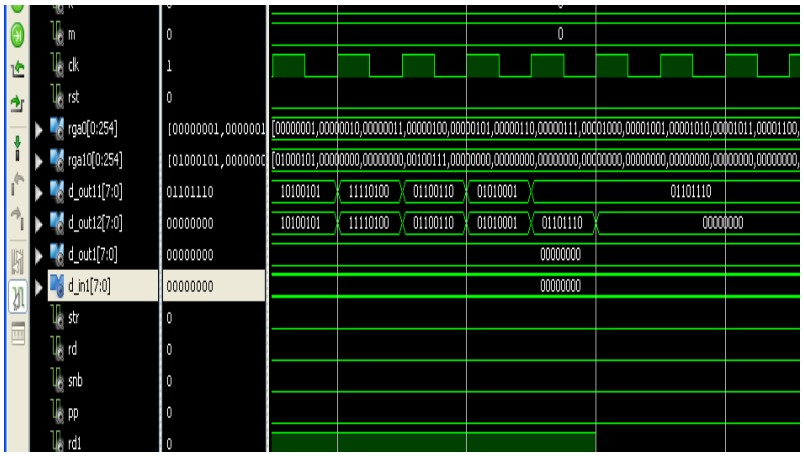


***Fig.4*.1.Input data of order (n,k) is (255,231)**

Fig4.1. Shows the input data sequence in the order of (255,231) which is giving to the encoding input

4.2. Encoded Data:

The code word generated by product reed Solomon encoder is given below as shown in fig. 4.2 after adding the parity symbols (255,247).

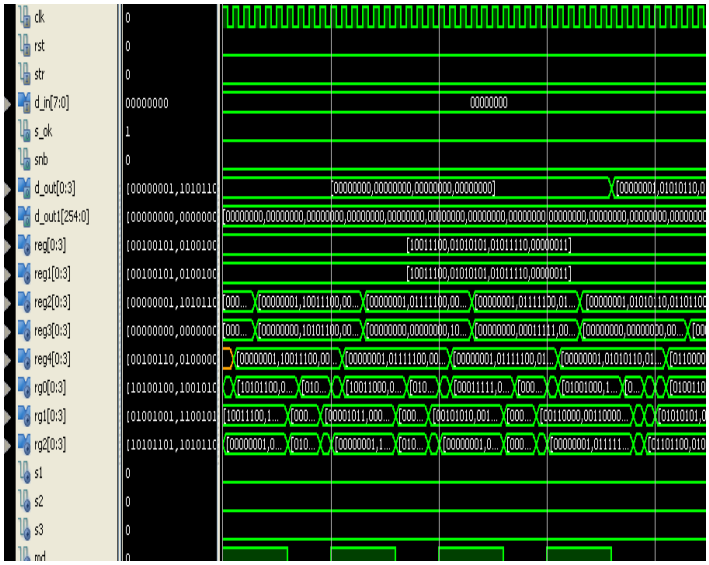


Fig.4.2.Encoder Output

PART-2 Decoder

4.3. Syndrome

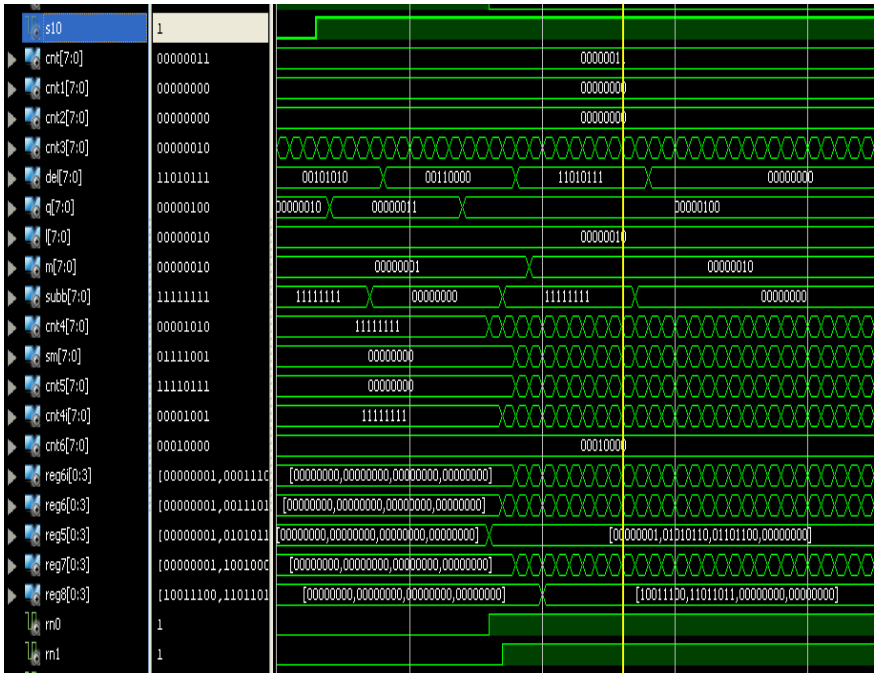


Fig.4.3. Syndrome Output

Calculating the Syndrome is the first step in decoding part and by using syndrome we can decide whether the error is present or not.The syndrome for the received data is as shown in fig.4.3

4.4. Decoder Error value:

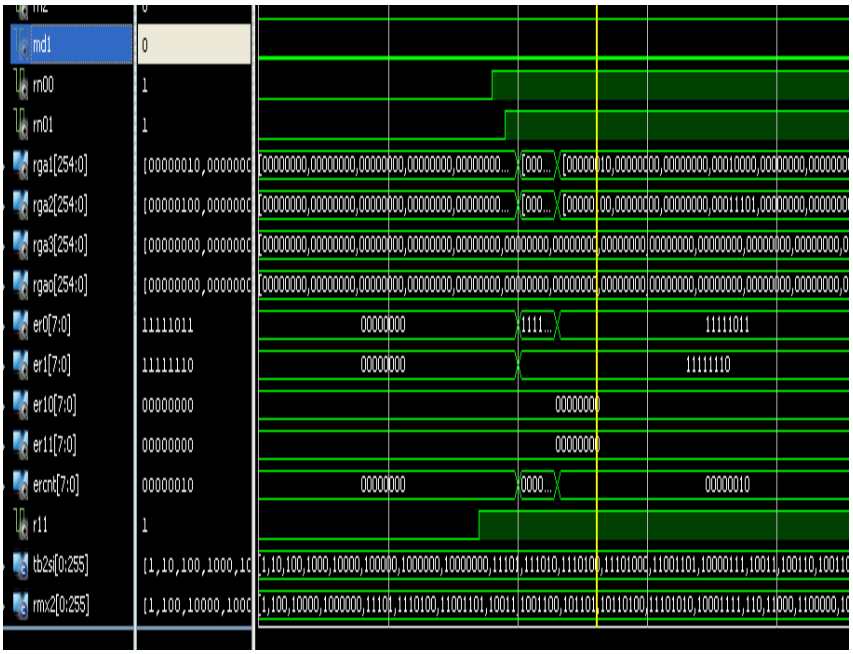


Fig.4.4. Decoder Error value

It shows the Magnitude of the error value that is nothing but identification of error position.

4.5. Error Correction

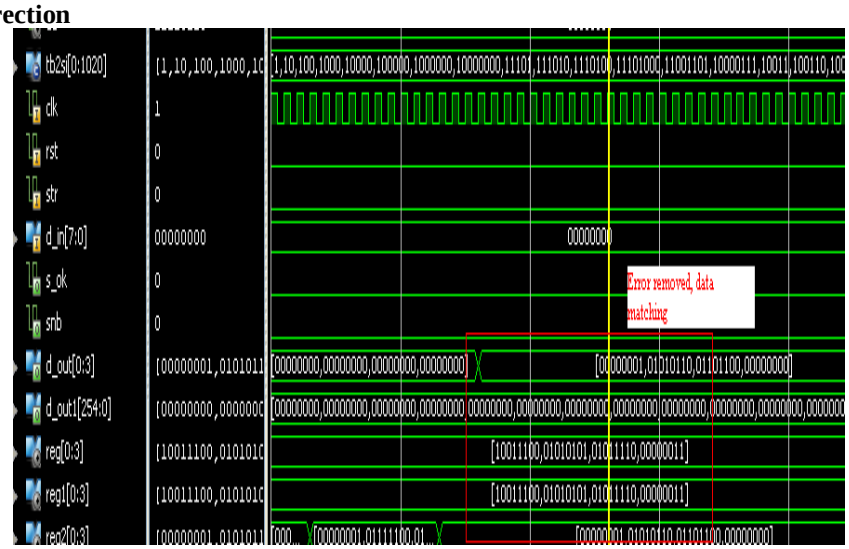


***Fig.4.5.* Error Correction**

This figure shows removing the errors and matching corrected data with the transmitted dada.

V. Conclusion

This note has described the theory and methods of Reed-Solomon coding and decoding with a view to facilitating implementation in dedicated hardware. This has particular relevance in this project proposes a (255, 231) 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 i,e. Product Reed-Solomon code can improve the gain, bandwidth and also power consumption. Implementations of standard, Reed-Solomon coding circuits for DVB-T have already been implemented in hardware as part of the Digital Radio Camera project. For the decoder, the techniques described have been tested in software for the DVB-T standard and in hardware for simpler codes. It is believed that this has identified many of the pitfalls involved in these processes.

References

[1] S. P. Kang, C. G. Kim, S. W. Rhee, and Y. Jee, "ASIC Implementation of Reed-Solomon Error Correction Circuits for Low Area Overhead on Memory System," International Cnference on Electronics, Information, and Communication (ICEIC 2008), pp. 339-342, June. 2008.

[2] B. Chen and X. Zhang, "Error Correction for Multi-Level NAND Flash Memory Using Reed-Solomon Codes," IEEE Workshop on Singal Processing Systems (SiPS), pp. 94-99, Oct. 2008.

[3] R. Micheloni, R. Ravasio, A. Marelli, et. al, "A 4Gb 2b/cell NAND Flash Memory with Embedded 5b BCH ECC for 36MB/s System Read Throughput," IEEE Inter. Solid-State Circuits Conf., Feb. 2006.

[4] F. Sun, K. Rose and T. Zhang, "On the Use of Strong BCH Codes for Improving Multilevel NAND Flash Memory Storage Capacity," IEEE Workshop on Signal Processing Systems (SiPS), Oct. 2006

[5] W. Liu, J. Rho and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND Flash memories," IEEE Workshop on Signal Processing Systems (SIPS), pp. 248- 253, 2006.

[6] R. Micheloni, R. Ravasio, and A. Marelli, et. al, "A 4Gb 2b/cell NAND Flash Memory with Embedded 5b BCH ECC for 36MB/s System Read 284Throughput," IEEE Inter. Solid-State Circuits Conf, pp. 497-506 Feb. 2006.

[7] F. Sun, K. Rose, and T. Zhang, "On the Use of Strong BCH Codes for Improving Multilevel NAND Flash Memory Storage Capacity," IEEE Workshop on Signal Processing Systems (SiPS), Oct. 2006.

[8] W. Liu, J. Rho, and W. Sung, "Low-Power High-Throughput BCH Error Correction VLSI Design for Multi-Level Cell NAND Flash Memories," IEEE Workshop on Signal Processing Systems (SiPS), pp. 248-253, 2006.

[9] K. Viswajith Seth, K.N. Srinivasan and S. Kamakoti, "Ultra folded high-speed architectures for Reed Solomon decoders," VLSI Design'06, pp.1063-9667, Jan. 2006.

[10] 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.

[11] S. Lin and D. J. Costello. Jr, Error Control Coding: Fundamentals and Applications, 2nd ed., Prentice Hall, 2004.

[12] S. Lin and D. J. Costello, Jr. Error Control Coding: Fundamentals and Applications (2nd edition), Prentice Hall, 2004.

[13] Y. Chen and K. K. Parhi, "Small area parallel Chien search architectures for long BCH codes," IEEE Trans. on VLSI, Vol 12, Issue 5, May 2004.

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

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

[16] L. Song, M. Yu and M. S. Shaffer, "10- and 40-Gb/s Forward Error Correction Devices for Optical Communications," IEEE Journal of Solid- State Circuits, Vol. 37, No. 11, Nov. 2002.

[17] 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.

[18] D. V. Sarwate and N. R. Shanbhag, "High-Speed Architectures for Reed-Solomon Decoders," IEEE Trans. On VLSI, vol 9, Oct. 2001.

[19] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, Wiley, Jan. 1999.

A.Vijaya Lakshmi , received B.E in Electronics and Communication Engineering and M.E. in Digital Systems Engineering from Osmania University. Presently working as an Associate Professor in Vardhaman College of Engineering, Shamshabad, Hyderabad.