FPGA Implementation of Improved RS Decoder using Uniform Syndrome Computation

SATYA SRAVANI SADHU¹, J. PREM KUMAR², L. KAVYA³

¹PG Scholar, Dept of ECE, Sridevi Women’s Engineering College, JNTUH, Telangana, India, Email: srav.sadhu@gmail.com.
²Assoc Prof, Dept of ECE, Sridevi Women’s Engineering College, JNTUH, Telangana, India, Email: yprems@yahoo.co.in.
³Asst Prof, Dept of ECE, Sridevi Women’s Engineering College, JNTUH, Telangana, India, Email:kavya.lakkakula@gmail.com.

Abstract: Many digital signaling applications in broadcasting use forward Error correction, a technique in which redundant information is added to signal to allow the receiver to detect and correct errors that may have occurred in transmission. Many different types of codes have been devised for this purpose but Reed-Solomon (RS) codes have proved to be good compromise between efficiency and complexity. The RS codes are widely employed as the error control code in numerous digital communication and storage systems. A Reed-Solomon encoder takes a block of digital data and adds extra redundant bits. RS decoder processes each block and attempts to correct errors and recover original data. The error control capability of these codes was improved by using many decoding algorithms, foremost being Berlekamp decoding algorithm. RS decoder determines the location and magnitude of errors in received polynomial caused due to noise while communication. For this, efficient decoding techniques like chien and Forney algorithms are used by the decoder. An improved RS decoder is proposed in this paper, in which a uniform syndrome computation technique is used. In this technique, syndromes of one test vector can be obtained from syndromes of previous test vector. The proposed decoder is modeled using VHDL. Simulated and synthesized in Xilinx 9.1i tool. The delay obtained in this decoder is proved to be less compared to prior designs.

Keywords: Reed-Solomon (RS) codes, RS Decoder, Unified Syndrome Computation (USC).

I. INTRODUCTION

Wireless technology is fast becoming a trend in present communication systems, as the demand for greater bandwidth allocation is being addressed by fixed wireless broadband access. However, the use of free space, as a medium, introduces many sources of error in the transmission of data across the channel. Burst Error (contiguous errors in the bit stream) is a common occurrence in digital communication systems, broadcasting systems and digital storage devices. Forward error correction is a technique in which redundant information is added to the original message, so that some errors can be corrected at the receiver, using the added redundant information. Reed Solomon Encoder and Decoder falls in the category of forward error correction encoders and it is optimized for burst errors rather than bit errors. Reed-Solomon (RS) codes are widely employed as the error control code in numerous digital communication and storage systems. Berlekamp developed the first practical decoding procedure for RS codes in 1968 [1]. In recent years, the error control capability was improved by Koetter and Vardy [2], by incorporating the reliability information from the channel into the algebraic soft-decision (ASD) decoding process. In the present Digital Communication systems, it is highly possible that the data or message may get corrupted during transmission and reception through a noisy channel medium. The environmental interference and the physical defects in the medium are the main causes for the data or message corruption in the communication medium, which leads to the injection of random bits into the original message and corrupt the original message. To have a reliable communication through noisy medium error correcting codes (ECC) [1], [2], [3], [4] has to be used.

The error correction is based on mathematical formulas, which are used by error correcting codes (ECC). Error correction takes place by adding parity bits to the original message bits during transmission of the data. Because of the addition of parity bits to message bits, the size of the original message bits becomes longer. Now this longer message bits is called “Codeword”. This codeword is received by the receiver at the destination, and could be decoded to retrieve the original message bits [5], [6]. Reed Solomon Codes are used for both encoding and decoding purposes. These codes are defined as RS (n, k) with m bit symbols. The RS codes detect and correct the errors in the symbols. Reed Solomon codes follow the Galois Field (GF) mathematical properties for encoding and decoding techniques [7], [8]. In this paper, an improved RS decoder is proposed in which a uniform syndrome computation technique is used. The proposed decoder is modeled using VHDL. Simulated and synthesized in Xilinx 9.1i tool. The delay obtained in this decoder is proved to be less compared to prior designs.
II. GALOIS FIELD

Galois field theory plays main role in the Reed Solomon Encoding and Decoding process. A Galois field consist of set of elements (numbers). The elements are based on a primitive element, usually denoted as α and take the values, (0, 1, α, α²,..., α^{N-1}) to form a set of 2ᵐ elements, where N =2ᵐ -1. field is then known as GF(2ᵐ). The value of α is usually choosing to be 2. The Reed-Solomon code is defined in the Galois field which contains a finite set of elements where any arithmetic operations on elements goes on such elements belonging to the same set. Burst errors are efficiently corrected by using cyclic codes. The Galois field or the Finite Fields are extensively used in the Err - Correcting Codes (ECC). In this section these finite fields are discussed. The Galois Field is a finite set of elements which has defined rules for arithmetic. These roots are not algebraically different from those used in the arithmetic with ordinary numbers but the only difference is that there is only a finite set of elements involved. They have been extensively used in Digital Signal Processing (DSP), Pseudo- Random Number Generation, Encryption and Decryption protocols in cryptography. A Finite Field is a field with a finite field order (i.e., number of elements), also called a Galois field. The order of a limited field is always a prime or a power of a prime. For every prime power, there exists exactly one finite field GF (pm). A Field is said to be infinite if it consists of infinite numbers.

The finite field or GF contains (2ᵐ –1) non zero elements. All finite fields contain a zero element and an element, called a generator or primitive element α, such that every non-zero element in the field can be expressed as a power of this element. Encoders and decoders for linear block codes over GF (2ᵐ), such as Reed- Solomon code s, require arithmetic operations in GF (2ᵐ). In GF (2ᵐ) addition and subtraction are simply bitwise exclusive-or. Multiplication can be performed by several approaches, including bit serial, bit parallel (combinational), and software. Division requires the reciprocal of the divisor, which can be computed in hardware using several methods, including Euclid’s algorithm, lookup tables, exponentiation, and subfield representations. With the exception of division, combinational circuits for Galois field arithmetic are straightforward. Fortunately, most decoding algorithms can be modified. In GF (2ᵐ) fields, there is always a primitive element α , such that you can express every element of GF(2ᵐ) except zero as a power of α . You can generate ever field GF (2ᵐ) using a primitive polynomial over GF(2ᵐ) field is modulo arithmetic. The simplest example of a finite field is the binary field.

Properties of Galois Field:
The main properties of a Galois fields are:

• A finite field or Galois field must be based on a prime number to ensure that each row and column of its addition and multiplication tables contains unique values.
• The primitive (root) of the field is used to derive all of the values in the field.
• A finite field or Galois field is indeed finite, but the field repeats an infinite number of times.
• The modulus of the field determines the values of its elements.
• The result of adding or multiplying two elements from the Galois field must be an element in the Galois field.
• Identity of addition —zero must be exist, such that a + 0 = a for any element a in the field.
• Identity of multiplication —one must be exist such that a * 1 = a for any element a in the field.
• For every element a in the Galois field, there is an inverse of addition element b such that a+b = 0. This allows the operation of subtraction to be defined as addition of the inverse.
• For every non-zero element b in the Galois field, there is an inverse of multiplication element b^{-1} such that bb^{-1}= 1. This allows the operation of division to be defined as multiplication by the inverse.
• Both addition and multiplication operations should satisfy the commutative, associative, and distributive law.

III. REED SOLOMON CODES

A. Introduction

Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications and storage. Reed-Solomon codes are used to correct errors in many systems including Storage devices (including tape, Compact Disk, DVD, barcodes, etc), Wireless or mobile communications (including cellular phones, microwave links, etc), Satellite communications, Digital television/DVB. A typical system is shown in figure 1:

Fig1. System employing RS decoder and Encoder

The Reed-Solomon encoder takes a block of digital data and adds extra "redundant" bits. Errors occur during transmission or storage for a number of reasons (for example noise or interference, scratches on a CD, etc). The Reed-Solomon decoder processes each block and attempts to correct errors and recover the original data. The number and type of errors that can be corrected depends on the characteristics of the Reed-Solomon code.

B. Properties of Reed-Solomon codes

Reed Solomon codes are a subset of BCH codes and are linear block codes. A Reed-Solomon code is specified as RS(n,k) with s-bit symbols. This means that the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol codeword. There are n-k parity
symbols of $s$ bits each. A Reed-Solomon decoder can correct up to $t$ symbols that contain errors in a codeword, where $2t = n-k$. The following diagram shows a typical Reed-Solomon codeword (this is known as a Systematic code because the data is left unchanged and the parity symbols are appended).

**Fig2. Reed Solomon code word**

**Example:** A popular Reed-Solomon code is RS (255,223) with 8-bit symbols. Each codeword contains 255 code word bytes, of which 223 bytes are data and 32 bytes are parity. For this code:

$$n = 255, \quad k = 223, \quad s = 8$$

$$2t = 32, \quad t = 16$$

The decoder can correct any 16 symbol errors in the code word: i.e. errors in up to 16 bytes anywhere in the codeword can be automatically corrected. Given a symbol size $s$, the maximum codeword length ($n$) for a Reed-Solomon code is $n = 2^s - 1$. For example, the maximum length of a code with 8-bit symbols ($s=8$) is 255 bytes. Reed-Solomon codes may be shortened by (conceptually) making a number of data symbols zero at the encoder, not transmitting them, and then re-inserting them at the decoder.

**IV. PROPOSED ARCHITECTURE**

**A. Improved Rs Decoder Using Uniform Syndrome Computation**

The architecture of proposed decoder consists of syndrome calculation architecture and syndrome update architecture. The block diagram of reduced-complexity decoder is given in Fig3. For the syndrome calculation architecture, it is the same as ordinary syndrome architecture. The syndromes of first test vector are calculated and later pass the syndrome update module in order. The Syndrome update module works simultaneously with key equation solver module. Since the KES used Reduced Inversion-less Berklemap Massey algorithm. It requires $2t$ clock periods to get the error locator and the evaluation polynomial, the syndrome update module has to finish the syndrome update in $2t$ clock periods to provide the syndromes of next test vector to the KES. The advantage of RiBM is that in each step the KES can be implemented simultaneously. Consequently, there is no free clock period for each unit in KES module. It provides higher throughput and can match the speed of syndrome update perfectly. The polynomial selection is based on Chien search algorithm. To reduce the extra storage for all error locators and evaluation polynomials, the Chien search architecture is employed to find the root number of error locator polynomial. Once the location of the error has been determined, Forney algorithm is used to evaluate the evaluator polynomial at the root. It uses the result to calculate the value of the error at the given location. Once this has been determined, the value is added to the received word to recover the corrected data. The addition occurs after the Error Locator polynomial evaluates to zero.

**Fig3. Architecture of Proposed RS decoder.**

**B. Syndrome Calculation**

The first step in decoding the received symbol is to determine the data syndrome. Here the input received symbols are divided by the generator polynomial. The result should be zero. The parity is placed in the codeword to ensure that code is exactly divisible by the generator polynomial. If there is a remainder, then there are errors. The remainder is called the syndrome. The syndromes can then be calculated by substituting the $2t$ roots of the generator polynomial $g(x)$ into $R(x)$. The syndrome polynomial is generally represented as,

$$S(x) = S_0 + S_1 x + S_{2t-1} x^{2t-1} = \sum_{j=0}^{n-1} \eta \alpha^j$$

Where, $\alpha$ is the primitive element. The basic syndrome calculation architecture is shown in fig4.
The syndromes of received codeword are computed as
\[ S_j = \sum_{l=0}^{n-1} \eta_l \cdot \alpha^{ij}, \text{ for } j=1 \text{ to } 2t \]
Where \( \alpha \) is the distinct element of GF \((2^q)\). For LCC decoding the syndromes of all \(2^n\) test vectors are required. For \(t^{\text{th}}\) test vector, the symbol in position \(i\), \(i \in C\) can be \( \eta_l \)
and this is denoted as \( S_{j,t} \),
\[ S_{j,t} = \sum_{l=0}^{n-1} \eta_l \cdot \alpha^{ij}, \text{ for } j=1 \text{ to } 2t, \tau=1 \text{ to } 2^n \]
If inputs are \( r_1, r_2 \)
For \( j=1 \text{ to } 2t, i=1 \text{ to } n-1, \tau=1 \text{ to } 2t \)
begin
\[ S_{j,1} = r_1 \cdot \alpha^{ij} \]
\[ \tau=\tau+1 \]
end
then for \( \tau=2 \text{ to } 2^n \)
\[ S_{j,\tau} = S_{j,(\tau-1)} + (r_1 + r_2) \cdot \alpha^{ij} \]
end

Outputs: \( S_{j,1}, S_{j,2}, S_{j,3}, \ldots, S_{j,2^n} \)

If all the test vectors are arranged in an order in which adjacent two vectors only have one different symbol, the syndromes can be calculated directly from the previous result. For the first test vector which selects all \( r_i \), \( i \in C \)
\[ S_{j,1} = \sum_{l=0}^{n-1} \eta_l \cdot \alpha^{ij}, \text{ for } j=1 \text{ to } 2t \]
For the rest test vectors, since there is only one different symbol in adjacent two test vectors, the syndromes of \( t^{\text{th}}\) test vector can be calculated from the result of \((\tau-1)^{\text{th}}\) test vector
\[ S_{j,1} = S_{j,(\tau-1)} + (r_1 + r_2) \cdot \alpha^{ij}, \text{ for } j=1 \text{ to } 2t \]
\[ S_{j,1} = S_{j,(\tau-1)} + \sum_{l=0}^{n-1} \eta_l \cdot \alpha^{ij}, \text{ for } j=1 \text{ to } 2t \]
\[ S_{j,\tau} = S_{j,(\tau-1)} + \sum_{l=0}^{n-1} (\eta_l - \eta_{l,(\tau-1)}) \cdot \alpha^{ij} \]
There is only one different symbol between the \((\tau-1)^{\text{th}}\) and the \(t^{\text{th}}\) test vectors and its position is denoted as \( C \)
\[ S_{j,\tau} = S_{j,(\tau-1)} + \sum_{l=0}^{n-1} (\eta_l - \eta_{l,(\tau-1)}) \cdot \alpha^{ij} \]
The uniform syndrome algorithm only requires calculating the syndromes for first test vector. Syndromes of other test vectors can be updated based on the result of first test vector by adding the syndrome difference.

C. Determination of Error-Locator Polynomial

The next step, after the computing the syndrome polynomial is to calculate the error values and their respective locations. This stage involves the solving of the \(2t\) syndrome polynomials, formed in the previous stage. These polynomials have \( v \) unknowns, where \( v \) is the number of unknown errors prior to decoding. If the unknown locations are \( (1,2,\ldots,i_v) \), the error polynomial can be expressed as,
\[ E(x) = Y_1x^{i_1} + Y_2x^{i_2} + \ldots + Y_vx^{i_v} \]
Where \( Y_i \) is the magnitude of the \(i^{\text{th}}\) error at location \( i \), \( Y_i \) is the field element associated with the error location \( i \), then the syndrome coefficients are given by
\[ S_j = \sum_{l=0}^{v} Y_l x^{i_l} \]
Where, \( j=1,2,\ldots,2t \). And \( Y_j \) is the error value and \( X_j \) is the error location of the \(j^{\text{th}}\) error symbol. The expansion of (4.8) gives the following set of \(2t\) equations in the \(v \) unknown error locations \( X_1, X_2, \ldots, X_v \) and \( v \) unknown error magnitudes \( Y_1, Y_2, \ldots, Y_v \)
\[ S_1(x) = Y_1X_1 + Y_2X_2 + \ldots + Y_vX_v \]
\[ S_2(x) = Y_1X_1^2 + Y_2X_2^2 + \ldots + Y_vX_v^2 \]
\[ \ldots \]
\[ S_{2t}(x) = Y_1X_1^{2t} + Y_2X_2^{2t} + \ldots + Y_vX_v^{2t} \]
The above set of equations must have at least one solution because of the way the syndromes are defined. This solution is unique. Thus the decoder's task is to find the unknowns given the syndromes. This is equivalent to the problem in solving a system of non-linear equations. Clearly, the direct solution of the system of nonlinear equations is too difficult for large values of \(v\). Instead, intermediate variables can be computed using the syndrome coefficients \( S_j \) from which the error locations, \( X_1, X_2, \ldots, X_v \), can be determined.
The polynomial is defined with roots at the error locations \(-1 \ i.e \ X_l-1 \) for \( l=1,2,\ldots,v \). The error location numbers \( l \), \( X \) indicate errors at locations \( i_l \) for \( l=1,2,\ldots,v \). This can be written as
\[ \sigma(x) = (1-xX_1)(1-xX_2) \ldots (1-xX_v) \]

E. Berlekamp- Massey Algorithm

The Berlekamp-Massey algorithm relies on the fact that the matrix equation of is not arbitrary in its form, rather, the matrix is highly structured. This structure is used to obtain the vector \( \sigma \) by a method that is conceptually more complicated. If the vector \( \sigma \) is known, then the first row of the above matrix equation defines \( S_1+1 \) in terms of \( S_1, S_2, S_3,\ldots,S_v \). The second row defines \( S_2+2 \) in terms of \( S_2, S_3,\ldots,S_v+1 \) and so forth. This sequential process can be summarized by the recursive relation,
\[ S_j = -\sum_{\tau=1}^{t} \sigma_\tau S_{j-\tau} \]
For fixed \( \sigma \), this is equivalent to the equation of an autoregressive filter. It can be implemented as a linear-feedback shift register with taps given by the coefficients of \( \sigma \). Using this argument, the problem has been reduced to the design of a linear-feedback shift register that will consequently generate the known sequences of syndromes. Many such shift registers exist, but it is desirable to find the smallest linear-feedback shift register with this property. This will give the least-weight error pattern with a polynomial \( \sigma(x) \) of smallest degree \( v \). The polynomial of smallest degree \( v \) is unique, since the \( v \times v \) matrix of the original problem is invertible. Any procedure for designing the autoregressive filter is also a method for solving the
matrix equation for the $\sigma$ vector. The procedure applies in any field and does not assume any special properties for the sequence $S_1, S_2 \ldots S_t$. To design the required shift register, the shift register length $L$ and feedback connection polynomial $\sigma (x)$ must be determined. The $\sigma (x)$ is in following form,

$$\sigma (x) = \sigma_p x^p + \sigma_{p-1} x^{p-1} + \cdots + \sigma_1 x + 1$$

F. Error value computation - Forney algorithm

Once the errors are located, the next step is to use the syndromes and the error polynomial roots to derive the error values. Forney Algorithm is generally used for this purpose. It is an efficient way of performing a matrix inversion, and involves two main stages. First the error evaluator polynomial $\omega(x)$ is calculated. This is done by convolving the syndromes with the error polynomial $\sigma (x)$ (from the Euclid’s Algorithm result).

$$\omega(x) = S(x) \sigma(x) \text{mod} (x^p)$$

This calculation is carried out at each zero location, and the result thus arrived is then divided by the derivative of lambda. Each such calculation gives the error symbol at the corresponding location. The error magnitude at each error location $x'$ is given by

$$e_i = \frac{\omega(\alpha^i)}{\sigma'(\alpha^i)}$$

If the error symbol has any set bit, it means that the corresponding bit in the received symbol is at error, and must be inverted. To automate this correction process each of the received symbol is read again (from an intermediate store), and at each error location the received symbols XOR’ed with the error symbol. Thus the decoder corrects any errors as the received word is being read out from it.

G. Find an error locator polynomial

This can be done using the Berlekamp-Massey algorithm or Euclid’s algorithm. Euclid’s algorithm tends to be more widely used in practice because it is easier to implement. However, the Berlekamp-Massey algorithm tends to lead to more efficient hardware and software implementations. The flowchart of Berlekamp-Massey Algorithm is given in figure5.

G. Chien Search for Error Positions

Once the error locator $\sigma (x)$ and error evaluator $\omega(x)$ polynomials have been determined using the above techniques, the next step in the decoding process is to evaluate the error polynomial $\sigma (x)$, and obtain its roots. The roots thus obtained will now point to the error locations in the received message. RS decoding generally employs the Chien search scheme to implement the same. A number ‘n’ is said to be a root of a polynomial, if the result of substitution of its value in the polynomial evaluates to zero. Chien Search is a brute force approach for guessing the roots, and adopts direct substitution of elements in the Galois field, until a specific $i$ from $i=0, 1, \ldots, n-1$ is found such that $\sigma (\alpha^i) = 0$. In such a case $\alpha^i$ is said to be the root and the location of the error is evaluated as $\sigma (x)$. Then the number of zeros of the error locator polynomial $\sigma (x)$ are computed and are compared with the degree of the polynomial. If a match is found the error vector is updated and $\sigma (x)$ is evaluated in all symbol positions of the codeword. A mismatch indicates the presence of more errors than can be corrected.

V. SIMULATION RESULTS

Software tool used in the project is Xilinx ISE 9.1i. It provides Xilinx PLD designers with the basic design process using ISE 9.1i Xilinx ISE is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize their designs, perform timing analysis, examine RTL diagrams, simulate a design’s reaction to different stimuli, and configure the target device with the programmer. Using Xilinx one can

- “Create a New Project”
- “Create an HDL Source”
- “Design Simulation”
V. CONCLUSION AND FUTURE SCOPE

In this thesis, Reed-Solomon decoder modified architecture is presented. Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications and storage. This architecture helps to find the syndromes of next test vector are obtained from syndromes of previous test vector. The improved RS decoder, gives better performance and significantly more efficient. Results are simulated using Xilinx 9.1i tool and synthesis report is obtained. The delay obtained is 3.837ns. Future work may reduce the delay.

VII. REFERENCES

[1] Reduced-Complexity LCC Reed–Solomon Decoder Based on Unified Syndrome Computation Wei Zhang, Hao Wang, and Boyang Pan IEEE TRANSACTIONS ON Very Large Scale Integration (VLSI) Systems 1063–8210 2012 IEEE.
FPGA Implementation of Improved RS Decoder using Uniform Syndrome Computation


Author’s Profile:

Ms. Satya sravani sadhu has completed her B.Tech in ECE Department from Maheshwara Institute of Technology, JNTU Hyderabad. Presently she is pursuing her Masters in VLSI System Design in Sridevi Women’s Engineering College, Vattinagulapally, Gandipet, Hyderabad, India.

Mr. J.Prem Kumar has completed his B.Tech from SV University. He has received his M.Tech from JNTU Hyderabad. He is having an experience of 15 years in Academic, presently working in Sridevi women’s Engineering College as Associate Professor in ECE Department. His interested research areas include Communication.

Mrs. L.Kavya has completed her B.Tech from JNTU Hyderabad. She has received her Master’s Degree from Srinidhi Institute of Technology, Hyderabad. Presently she is working as Assistant Professor in Sridevi women’s Engineering College, Hyderabad. Her interested research areas include signal Processing.