IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 1, March, 2013
ISSN: 2320 - 8791
www.ijreat.org

Implementation Of A RS Decoder Architecture using Artificial Neural Network

[1]V.J.K.Kishor Sonti & [2]Mrinmoy Sandilya

**[1]Assistant Professor, Department of Electronic & Communication Engineering,
Sathyabama University
Chennai, India.**

**[2] Student, Department of Electronic & Communication Engineering,
Sathyabama University
Chennai, India.**

Abstract

In this paper, an architecture for a typical Reed Solomon Decoder using Artificial Neural Network is presented. Reed Solomon codes are extensively used burst error correcting codes, used in applications like error correcting of data in CD/DVD, error correcting of faulty data read out of a bar code, and also in fields like data transmission such as DSL and WiMAX and satellite transmission. These are cyclic BCH (Bose-Chodhuri-Hocquenghem) codes but unlike BCH codes Reed Solomon codes are non binary codes . The architecture of a typical Reed Solomon encoder and decoder uses concepts of Finite fields and Galois field operations from abstract algebra extensively. Choosing the right methods of realizing Galois field operations play a crucial role in efficient realization of a Reed Solomon Decoder hardware in terms of circuit complexity and time complexity.

This paper explores the possibility of incorporating the concepts of Artificial Neural Network in realization of Reed Solomon Decoder architecture in an attempt to reduce the complexity. An artificial neural network is a massively parallel network of artificial neurons capable of parallel computations. The implementation of syndrome block is done with HDL synthesis report (Macro statistics count of 4 3-bit registers, 121 XOR-2 gates) in Xilinx ISE project navigator.

Keywords - Reed Solomon, Galois field, Artificial Neuron, finite field, syndromes

1.Introduction

The Reed–Solomon code is a block code generally denoted as (n,k,d) codes where n is the codeword length, k is the message symbol length and d is the minimum distance between two code words, also interpreted as the number of places in which two code elements differ or the design distance. The encoder takes k data symbols of length s bit each and adds 2t parity symbols to make a codeword of length n. A linear code is defined as a subspace of a binary vector space. Subspaces have basis vectors which are usually rows of Generator matrix. If rank(G) = k, then we have a k dimensional space.

Subspace is of the type $[G_1\ G_2 \ldots\ldots G_n]$

So if we have a linear code defined by a k*n generator matrix G with rank k than we have k-dimensional vector subspace of F_2^n with basis as rows of G. Sum of two or more code words equals to another codeword , as the elements form a finite field they satisfy the axioms of finite field.

A set F with two binary operations + and * is called a field if

1. (F, +) is an Abelian Group, 0 is additive identity
2. (F-{0}, *) is an Abelian group, 1 is multiplicative identity
3. The group is closed under Associativity

If F is a finite field, a polynomial f(X) with coefficients in F is said to be irreducible over F if and only if f(X) is irreducible as an element of the polynomial ring over F (that is, in F[X]) and since the polynomial ring F[X] is a unique factorization domain, a polynomial f(X) is irreducible if and only if it is prime as an element of F[X].[1]

Table 1.1: Comparison Of BCH and RS codes [9]

Code	Block length	Primitive element	Parity check matrix	Generator polynomial
BCH	n	Ord $\alpha \geq n$	H	LCM($M_\alpha, M_\alpha^{2} \ldots M_\alpha^{d-1}$)
RS	n	$\alpha \in F_2^m$	H	$\prod_{i}^{d-1} (x+\alpha^i)$

If we start with a code word C (n,k) code with a generator matrix $G = [I_k\ P]$ then $C = m*G = m*[I_k\ P]$. We can write as

$$[\ \underbrace{C_1\ \ C_2\ \ C_3 \ldots\ldots C_k}_{\text{Message}}\ \ \underbrace{C_{k+1} \ldots\ldots C_n}_{\text{Parity}}] = [m_1\ m_2 \ldots\ldots m_k\ P]$$

$$\begin{pmatrix} C_{k+1} \\ | \\ C_n \end{pmatrix} = P^T \begin{pmatrix} C_1 \\ | \\ C_k \end{pmatrix} = [\ m\ P\ ]^T$$ .This can be written as

$$[\ P^T\ \ I_{n-k}\ ] \begin{pmatrix} C_1 \\ | \\ C_k \\ | \\ C_n \end{pmatrix} = \begin{pmatrix} 0 \\ | \\ 0 \\ | \\ 0 \end{pmatrix} \qquad (1)$$

So H denoted as parity check matrix is $H = [\ P^T\ \ I_{n-k}\ ]$

$$C = \{C : C = mG,\ m \in F_2^{\,k}\} = \{C : HC^T = 0,\ C \in F_2^{\,n}\ \} \qquad (2)$$

A dual coding can be defined as $C^{\perp}$ where

$C^{\perp}$ = Dual of C= $\{C: CU^{\perp} = 0$, for all $U \in C\}$. Here the row space of H is $C^{\perp}$ and it is also a subspace of $F_2^{\,n}$. It can also be defined as set of all vectors orthogonal to every vector in C. The following table shows a comparison between BCH and Reed Solomon Codes:

Where
$$H = \begin{pmatrix} 1\ \alpha\ \alpha^2\ \alpha^3 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \alpha^{n-1} \\ 1\ \alpha^2\ (\alpha^2)^2\ (\alpha^2)^3 \ldots\ldots\ldots\ (\alpha^2)^{n-1} \\ 1\ \alpha^3\ (\alpha^3)^2\ (\alpha^3)^3 \ldots\ldots\ldots\ (\alpha^3)^{n-1} \\ | \\ 1\ \alpha^{d-1}\ (\alpha^{d-1})^2\ (\alpha^{d-1})^3 \ldots\ldots (\alpha^{d-1})^{n-1} \end{pmatrix} \qquad (3)$$

d = Design distance

The cyclic property of RS codes is interpreted as if we have a codeword set C such as $C = [C_0\ C_1\ C_2 \ldots C_{n-1}]$: $HC^T = 0$ then this implies that $C^{/} = [C_0\ C_1\ C_2 \ldots C_{n-1}]$: $HC^{/T} = 0$ is valid only for $n = 2^{m-1}$.

For RS codes t is defined as the error correcting property for a (n,k) code and defined as 2t = n-k= Number of parity symbols. The generator polynomial is of the form

$$G(x) = G_0 + G_1 x + G_2 x^2 + \ldots\ldots\ldots + G_{2t-1} x^{2t-1} + x^{2t} \qquad (4)$$

Since the generator polynomial is of degree 2t, there must be precisely 2t successive powers of α that are roots of the polynomial. We denote the roots to be $\alpha, \alpha^2, \alpha^3 \ldots\ldots \alpha^{2t}$. That gives us the codeword polynomial to be sent over defined as

$x^{n-k}.m(x) = Q(x).G(x) + P(x)$ resulting codeword is

$$C(x) = P(x) + x^{n-k} .\ m(x). \qquad (5)$$

2. Galois field Adder and multiplier

Designing a Galois field adder and multiplier for the field operations depends heavily on the efficiency of the circuit. An example code using the primitive polynomial $x^3 + x + 1 = 0$ for the $GF(2^3)$ is shown. The look up tables shows the additions and multiplications performed in GF(8) with the specified primitive polynomial.

Table 2.1: Equivalent bit values

α^0	α^1	α^2	α^3	α^4	α^5	α^6
0	1	2	4	3	6	7

If a, b are input polynomials in the form $a_2x^2 + a_1x + 1$ and $b_2x^2 + b_1x + 1$ then $z = a*b$ can be obtained as the remainder after a long division with the primitive polynomial as

$Z_2 = a_2b_1 + a_0b_2 + a_1b_1 + a_2b_2$;

$z_1 = a_1b_0 + a_0b_1 + a_2b_2 + a_2b_1 + a_1b_2$;

$z_0 = a_0b_0 + a_2b_2 + a_1b_2$

Table 2.2 Look up table for multiplication

×	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	3	1	7	5
3	3	6	5	7	4	1	2
4	4	3	7	6	2	5	1
5	5	1	4	2	7	3	6
6	6	7	1	5	3	2	4
7	7	5	2	1	6	4	3

The corresponding bit values are multiplied with AND operation in VHDL and the addition is performed with XOR operation. To perform all the GF(8) operations a package has been created in VHDL which will perform addition, multiplication, inverse in GF(8).

3. Decoder Architecture

Now let us assume an error polynomial as $e(x) = \sum e_{n}\,.\,x^{n}$. Then in the decoder end, we receive R(x) = C(x) + e(x), where R(x) is the received polynomial which has an error of magnitude and value e(x) incorporated in it. The flowchart for a typical RS decoder is shown here. The first step is to calculate syndrome values. The syndromes are the results of a parity check performed on R(x) to determine whether R(x) € C.

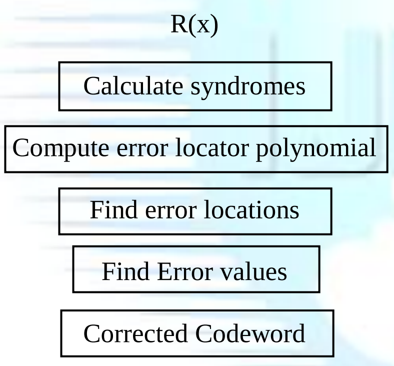

Figure 2.1 Decoder steps Flowchart

For calculating syndromes there are 2t equations.

$S_1 = r(\alpha)$, $S_2 = r(\alpha^2)$,……….$S_{2t} = r(\alpha^{2t})$. The non-zero values of syndromes conform that the received polynomial has some errors. Supposing there are v errors in locations x^{j1}, x^{j2}…x^{jv}, we formulate an error polynomial of the form

$$\begin{pmatrix} S_1 & S_2 & S_3\ldots & S_{t-1} & S_t \\ S_2 & S_3 & S_4\ldots & S_t & S_{t+1} \\ S_3 & S_4 & S_5\ldots & S_{2t-3} & S_{2t-2} \\ & | & & & \\ St_{-1} & S_{t+1} & S_{t+2}\ldots & S_{2t-2} & S_{2t-1} \end{pmatrix} \begin{pmatrix} \sigma_t \\ \sigma_{t-1} \\ \sigma_{t-2} \\ | \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} -S_{t+1} \\ -S_{t+2} \\ -S_{t+3} \\ | \\ -S_{2t} \end{pmatrix} \quad (6)$$

$e(x) = e_{j1}\, x^{j1} + e_{j2}\, x^{j2}\ldots\ldots e_{jv}\, x^{jv}$. Once a non zero syndrome vector has been computed, it's necessary to learn the location of error. So we define $\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x)\ldots.(1 + \beta_v x)$. This can be written as $\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + \ldots\ldots\ \sigma_v x^v$. The roots of σ(x) are $1/\ \beta_1$, $1/\ \beta_2\ldots 1/\ \beta_{v.}$ the reciprocals of the roots are error locator numbers of error pattern e(x).

The minus sign does not alter the value and can be regarded as plus as both minus and plus operations in modulo 2 are the same and denoted by XOR function. To decode the BCH codes, Elwyn Berlekamp proposed an algorithm in 1967 and in 1969 James Massey found its application to find the shortest linear feedback shift registers, and named the algorithm as "Berlekamp Iterative algorithm" [2]. Then the algorithm is known as Berlekamp-Massey algorithm (BM), it also finds the minimal polynomial of a linear sequence. After the error locator polynomial σ has been found, those values are passed to the Chien search and Forney algorithm blocks after which the corrected codeword is known.

4. RS decoder using Neural Network approach

The advantages of using neural network is that its based on parallel architecture that can give parallel results without performing the typical calculations. Its basically a mapping from inputs provided to the architecture, to the already stored corresponding outputs. Here the GF(8) operations can be so obtained in a look up table based approach which will immediately give the result. A single artificial neuron is the basic design element based on which we built up networks which contains an input layer, hidden layer(s) and an output layer. Figure 4.1 shows the implementation of an MP model of single Neuron with a step (hard limit) threshold function.

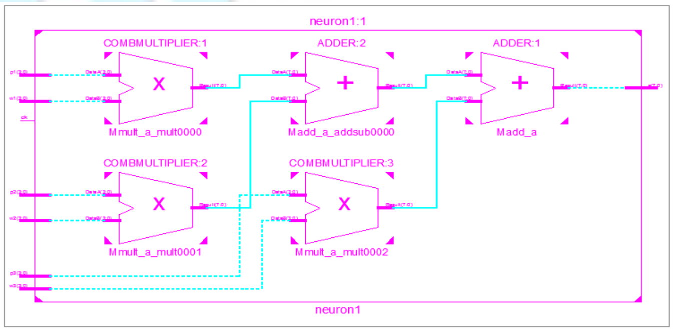

Figure 4.1 RTL schematic of MP model artificial Neuron with step threshold function

The single neuron can be incorporated in networks to realize rather complex functions. Figure 4.2 shows a NOR gate implementation of ANN.

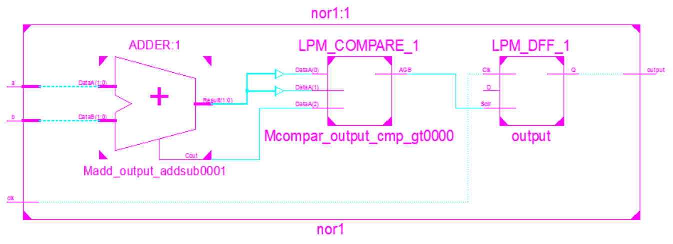

Figure 4.2 An artificial neural network NOR gate

The efficiency of a Neural Network depends on the choice of threshold function. The hard limit function is not so efficient from the real point of view of imitating a biological neuron. However the sigmoid function, traditionally used in ANNs, is not suitable for direct digital implementation as it consists of an infinite exponential series [1-2]. Thus most implementations resort to various methods of approximating the sigmoid function in hardware typically by using lookup tables (LUTs) to store samples of the sigmoid function for approximation. For avoiding complexity hard limit function is adopted here. In RS decoder implementation also a look up table based approach can replace GF(8) operations to avoid complexity. The behaviour is given by considering the topology of the neural network chosen and the activation function of neurons in each layer, the output depends on the training of the network from the given data in the decoding module. The proposed work is basically about designing an easy approach to calculate syndromes by not going the typical way of putting all the powers of α in the received polynomial to obtain syndromes but to a look up table based approach where data values of GF(8) operations are already stored. The higher values of α are given as weights to the unit which associates the weights to the already existing values and

Results

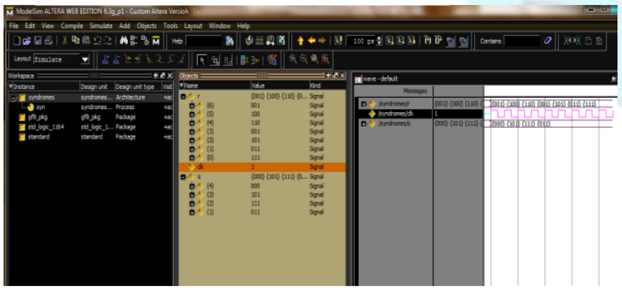

Figure 1: Syndrome Value calculation result

Figure 1 shows syndrome value calculations. The received polynomial input is r(x)= $\alpha^0+ \alpha^2x+ \alpha^4x^2+$

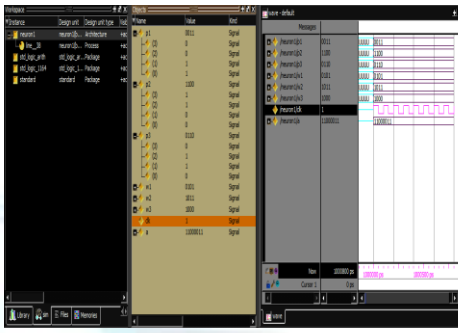

Figure 2: Artificial Neuron

Figure 2 shows an artificial neuron structure. The weights get mult
and and we get a final output as shown in the simulation result.

finally performs a sum. This reduces the time consumption in calculating the syndromes. The BM algorithm unit and the error value and magnitude unit are dependent on the syndrome values, so the vales can be fed parallel once they are calculated. For the parallel decoding algorithm is proposed to replace the iterative time for processing through an artificial neural network, it must be implemented in hardware. It have been used for ANN implementation due to accessibility, ease of fast reprogramming, and low cost, permitting the fast and non expensive implementation of the whole system.[5]

Figure 3: Top module entity simulation of RS decoder

Figure 3 shows the simulation result of the final RS decoder. For RS code r(x)= $\alpha^0+ \alpha^2x+ \alpha^4x^2+ \alpha^0x^3+ \alpha^6x^4+ \alpha^3x^5+ \alpha^5x^6$, we get corrected codeword c(x)= $\alpha^0+ \alpha^2x+ \alpha^4x^2+ \alpha^6x^3+ \alpha^1x^4+ \alpha^3x^{5+}\alpha^5x^6$

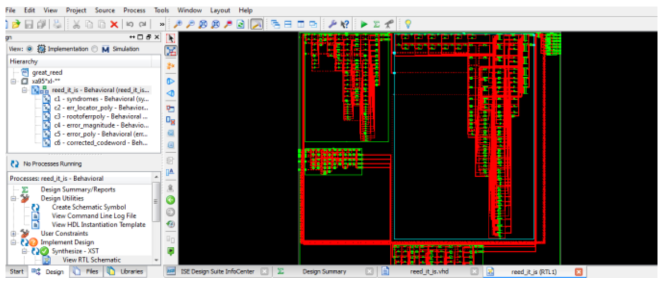
File
Edit
View
Project
Source
Process
Tools
Window
Layout
Help
View: Implementation
Simulation
Hierarchy
great_reed
xa95*xl-**
reed_it_is - Behavioral (reed_it_is...
c1 - syndromes - Behavioral (sy...
c2 - err_locator_poly - Behavior...
c3 - rootoferrpoly - Behavioral ...
c4 - error_magnitude - Behavio...
c5 - error_poly - Behavioral (err...
c6 - corrected_codeword - Beh...
No Processes Running
Processes: reed_it_is - Behavioral
Design Summary/Reports
Design Utilities
Create Schematic Symbol
View Command Line Log File
View HDL Instantiation Template
User Constraints
Implement Design
Synthesize - XST
View RTL Schematic
Start
Design
Files
Libraries
ISE Design Suite InfoCenter
Design Summary
reed_it_is.vhd
reed_it_is (RTL1)

Figure 4: RTL synthesis of RS

5.Conclusion

The paper has been implemented using Galois Field[8] Reed Solomon codes and the operations can be easily extended to GF[256] codes. The Galois field operations are subjected to irreducible polynomial x^3+x+1. Subjecting Galois field operations to another irreducible polynomial will change the results. The work has been carried in Xilinx ISE project navigator (Release version 12.3; Application Version M.70d). The results obtained are in accordance with theoretical facts.

6.References

[1] M. E. Buckley and S. B. Wicker, "Neural network for predicting decoder error in turbo decoders," *IEEE Communications Letters*, vol. 3, no.5, pp. 145–147, 1999
[2] J. J. Blake, L. P. Maguire, T. McGinnity, and L.J. McDaid, "Using Xilinx FPGAs to implement neural networks and fuzzy systems," *IEE Colloquium on Neural and Fuzzy Systems: Design, Hardware and Applications*, vol. 133, pp. 1/1–1/4, 1997

[3] Sandoval Ruiz, Cecilia,"FPGA prototyping of Neuro-Adaptive Decoder", University of Carabobo

Telecommunication

[4] Mukhtar, H, "Reed-Solomon encoder/decoder application using a neural network" (Proceedings Paper), 1991

[5] X. Yu and D. Dent, "Implementing neural networks in FPGAs," *IEE Colloquium on Hardware Implementation of Neural Networks and Fuzzy Logic*,vol. 61, pp. 1/1–1/5, 1994

[6] B. Ganesh, D. Muralidharan,"Efficient VLSI Architecture for Low Complexity Parallel Reed Solomon Decoder" European Journal of Scientific Research ISSN 1450-216X Vol.73 No.1 (2012), pp. 5-12

[7] Rafid Ahmed Khalil, Sa'ad Ahmed Al-Kazzaz, "Digital Hardware Implementation of Artificial Neurons Models Using FPGA"

[8] Reed-Solomon Codes by Bernard Sklar

[9] NPTEL lecture, Coding theory by Dr. Andrew Thangaraj, Dept. of Electronics and communication Engg., IIT Madras

[10] Hardware Implementation of Finite-Field Arithmetic, Mc Graw Hill, Jean-Pierre Deschamps, José Luis Imaña, Gustavo D. Sutter

[11] Compensatory Neurofuzzy Systems with Fast Learning Algorithms, Yan-Qing Zhang, Member, IEEE, and Abraham Kandel, Fellow, IEEE, IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 9, NO. 1, JANUARY 1998

[12] Reed Solomon Decoder: TMS320C64x Implementation Jagadeesh Sankaran Digital Signal Processing Solutions
[14] Field Programmable Gate Array Implementation of Reed-Solomon Code, RS(255,239)- Kenny Chung Chung Wai (Design Engineer Xelic Inc., Pittsford, New York 14534), Dr. Shanchieh Jay Yang(Assistant Professor R.I.T, Rochester, New York 14623)

[15] Application Note: CoolRunner-II CPLDs XAPP371 (v1.0) September 26, 2003 CoolRunner-II CPLD Galois Field GF(2m) Multiplier