FPGA prototyping of Neuro-Adaptive Decoder

SANDOVAL RUIZ, CECILIA
Attached investigator of IMYCA (Institute Mathematical and Applied Computing)
University of Carabobo
Telecommunication Department, UNEFA
Carretera Nacional Maracay-Mariara, frente a Base Sucre, Edo-Aragua
VENEZUELA
csandoval1@uc.edu.ve; cecisandova@yahoo.com

Abstract: - In this paper we present a design of a neuro-adaptive decoder with Very High Speed Integrated Circuit Hardware Description Lenguage (VHDL) based on Multi-Layer Perceptron Neural Network (MLP), FPGAs have been used for ANN implementation due to accessibility. For the design to choice case studie Reed Solomon RS(7,4) code, the methodology correspond to modular design of Artificial Neural Network, for the configuration of network elements in VHDL was used Xilinx ISE 11 tool. In the results presents structure of ANN, report of training, VHDL codes of ANN Decoder, the Register Transfer Level (RTL) schematic and synthesis report.

Key-Words: - ANN, ECC Decoder, Reed Solomon, FPGA, VHDL, modular design

1 Introduction

In the last years, there has been a huge trend towards building intelligent systems that are inspired by biological systems from nature. Many nature-inspired algorithms and approaches have been developed and have become popular for designing intelligent systems with application to various real-world problems [1]. The biologically inspired neural models generally rely on massive parallel computation. Thus the high-speed operation in real-time applications can be achieved only if the networks are implemented using parallel hardware architectures [2].

In [3] present a recurrent type neural networks (RNN) that have been successfully used in various fields of digital communications primarily due to their nonlinear processing, possible parallel processing that could accommodate recent requirements for high-speed signal transmission and, also, expected efficient hardware implementations [4]. For the RNN decoder to be of any real practical use, it must have a hardware realization that offers some benefits in terms of decoding speed, ease of implementation, or hardware complexity. The hardware implementation of artificial neural networks has been an active area of research. As techniques for implementing neural networks evolve, the RNN decoder, which has already shown to be competitive at an algorithmic level, may become a viable option in practical implementations [3].

This investigation we have oriented to ECC (Error Corrector Code) using Multi-Layer Perceptron Neural Network (MLP) with implement based on hardware. The main objective to investigate Hardware implementation of the decoding algorithm based on ANN application using FPGA technology. The choice of target technology is FPGAs, due to being capable of exploiting the parallelism inherent to the ANN decoder. 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 nonexpensive implementation of the whole system [2].

In [5-7] has presented an encoder - decoder RS (7.4) implemented on hardware, in which it was found that the decoding application *n* clock cycles to be implemented through an iterative decoding algorithm [8], where processing time is of great importance.

In [8] present as goal to propone an architecture that could be considered a general-purpose microprocessor for neurocomputing. This idea has been provided by the general to treat the task and in this regard is sought the design of a general- purpose neural network for channel decoding.

In this paper we present a design ANN decoder, we choice a case study: Reed-Solomon codes are widely used in deep-space communication, compact disc audio systems, and frequency-hopped systems, for this reason has been considered a source of interest for the project. Moreover, Neural Network implementation of these codes has resulted in reduced complexity, enhanced error correction capability, fast processing, and improved signal-to-noise ratio [4**],** therefore been considered as an object of study for this research, making use of VHDL hardware description language for parallel configuration**.**

ISBN: 978-960-474-257-8

1.1 Reed Solomon Codes

RS codes are a class of linear, non-binary, cyclic block codes. This class is a subfamily of the family BCH codes (Bose, Chaudhuri, Hocquenghem) are a class of linear and cyclic block codes that can be considered as a generalization of the Hamming codes [9], the selection of large block lengths as they can be designed for any value of the error-correction capability *t*. These codes are defined over the Galois field GF(*q*).

In order for the transmitted data to be corrected in the event that it acquires errors, it has to be encoded. The receiver uses the appended encoded bits to determine and correct the errors upon reception of the transmitted signal. The number and type of errors that are correctable depend on the specific Reed-Solomon coding scheme used [10].

Reed-Solomon codes are usually referred to as (*n*,*k*) codes, with *s*-bit symbols, where *n* is the total number of symbols in a code block and *k* is the number of information or data symbols. In a systematic code, the complete code block is formed from the *k* data symbols, followed by the *n-k* check symbols [10]. Where, the maximum number of symbol errors in a block that can be guaranteed to be corrected by the Reed-Solomon algorithm is $t = (n-k)/2$.

A Reed-Solomon code is also characterized by two polynomials: the field polynomial and the generator polynomial. The field polynomial defines the Galois field, of which the symbols are members. The generator polynomial defines how the check symbols are generated.

Coding Theory for RS codes

Let α be a primitive element in $GF(q^m)$ and let $n = q^m - 1$. Let $m = (m_o, m_1, \ldots, m_{k-1}) \in GF(q^m)^k$ be a message vector and let $m(x) = m_o + m_l x + \ldots + m_{k-1}x^{k-1} \in GF(q^m)[x]$ be its associated polynomial. Then the encoding is defined by the mapping $p : m(x) \rightarrow c$.

In constructing BCH codes, we looked for generator polynomials over *GF(q)* (the small field) so we dealt with minimal polynomials. Since the minimal polynomial for an element β must have all the conjugates of β as roots, the product of the minimal polynomials usually exceeds the number *2t* of roots specified [11].

The fig, 1 shows the architecture RS encoder, it present elements as shift register, xor gates, multiplexer and the multiplier in algebra of finite field, where the correspondent coefficients have been assigned for each multiplier over finite field; it has been configured in VHDL.

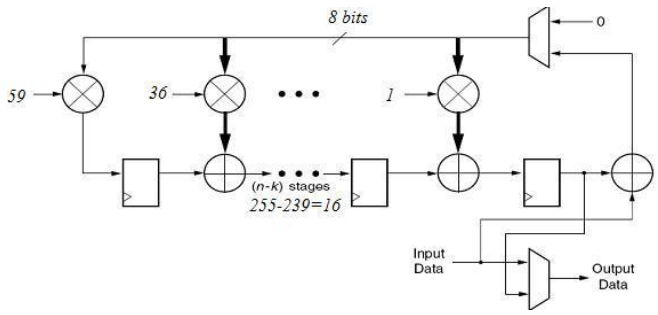

Fig. 1. Reed Solomon Encoder Architecture

2 FPGA Implementation of ANNs

FPGA-based ANNs can be tailored to specific ANN configurations; there is no need for worst-case fully interconnected designs as in full-custom VLSI [12].

In general, all ANN architectures consist of a set of inputs and interconnected neurons, this allows designing the network in a modular way through basic components. The neuron can be considered the basic processing element, and its design determines the complexity of the network. The neuron consists of three main elements: the synaptic connections, the adder, and the activation function.

The fundamental problem limiting the size of FPGA-based ANNs is the cost of implementing the multiplications associated with the synaptic connections because fully parallel ANNs require a large number of multipliers. Although prototyping itself can be accomplished using FPGAs which offer high number of multipliers, the overall goal of the decoder design is to obtain a highly parallel decoder and to use as few resources as possible [3].

The main problem in the field of error-correcting codes is to design good codes: codes that can correct many errors and whose encoding and decoding procedures are computationally efficient [13]. In the past several years substantial efforts have been made to apply RNNs in error control coding theory [3]. Initially, these networks were applied for block codes decoding [13, 14], then for convolutional [3] and turbo codes decoding [15]. This allows us to work on a generalization for different decoders through artificial neural networks.

The time demanded by the decoding algorithm for the different code is high, such as Reed Solomon codes, additionally the VLSI implementation of these codes is still very complex, and encoding/decoding by using these chips is very time consuming [4**]**, for which treatment is proposed decoding by parallel neural networks, with training in pre-execution time for optimizing the processing, because requires a decoder capable of processing over the transmission speed of the system.

ISBN: 978-960-474-257-8

3 Case Study: ANN Decoder design - Reed solomon (7,4)

For hardware implementation it is considered important to separate the learning and decodification phase of an ANN. Its implementation is essentially an implementation of a learning algorithm (gradient descent) [3].

The first step is to outline the response of the decoder to RS codes (7,4). Initially, we must model the architecture of the network MLP being one of the best universal approximate [14]. It in the figure 2 shows the flowchart for the network construction.

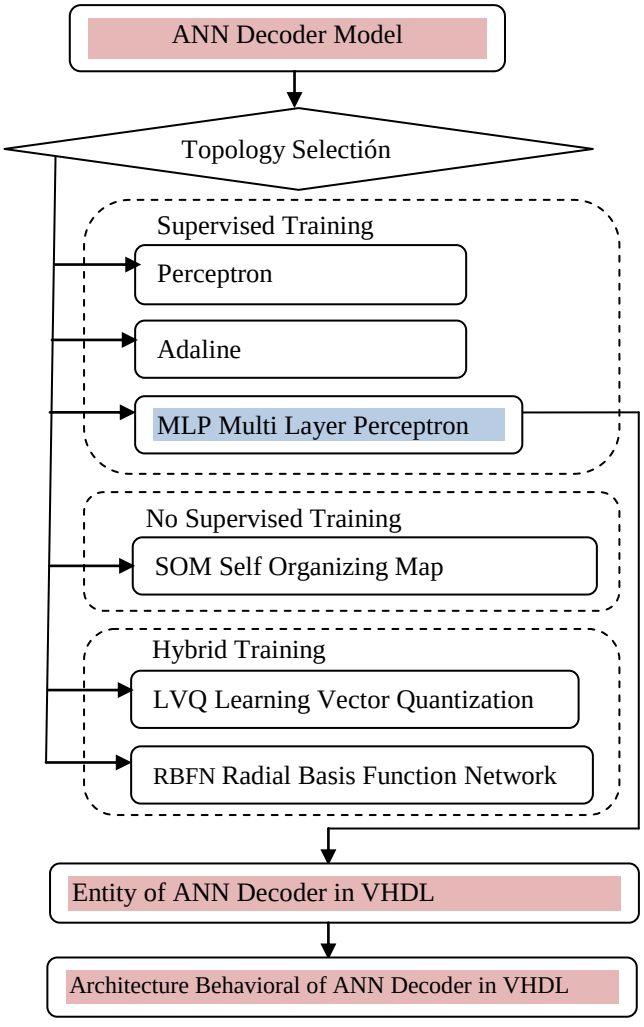

Fig. 2. Model of ANN

From the ANN model is required to build the network, for which the architecture is considered MLP and defining characteristics of each layer, the number of neurons in the input layer corresponds to the number of entries to the network, just as the number of neurons in the output layer matches the number of RS decoder outputs (7.4) of 7 inputs - 3 outputs.

The structured of the neural network corresponds to a MLP forward flow, with 3 layers, input layer presents the seven (7) input neurons, hidden layer are arranged four (4) neurons with sigmoid activation function and output layer are three (3) neurons with identity activation function, as shown in Figure 3.

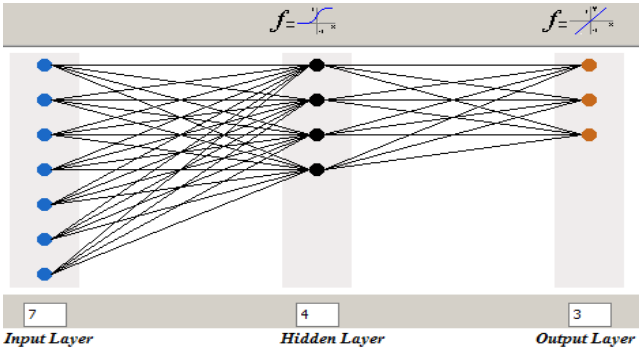

Fig.3. Estructure of ANN designed

Training Network

Once defined the structure of the network proceeded to load the input data and expected output for training the adaptive process, obtaining the graph of error in relation to the number of seasons as shown in Figure 4.

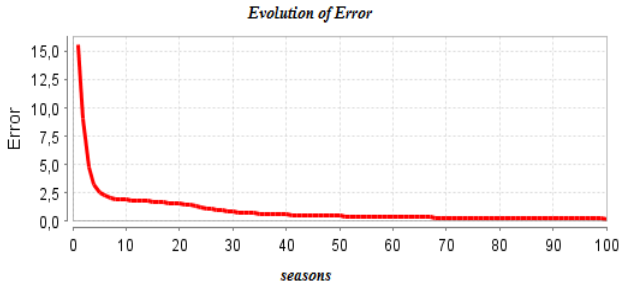

Fig. 4. Evolution of error

It asked the network to observe the response of the decoder as show in Figure 5.

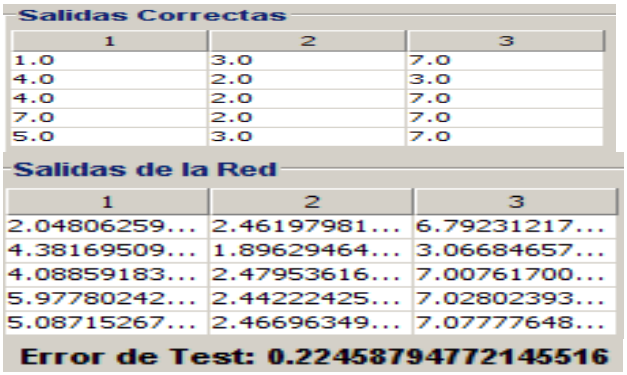

Fig. 5. Error of test

Get each of the weights and threshold parameters are characteristic for the decoding network. Figure 6

ISBN: 978-960-474-257-8

shows the weights of the output layer, these values must be adjusted in the network implemented on hardware.

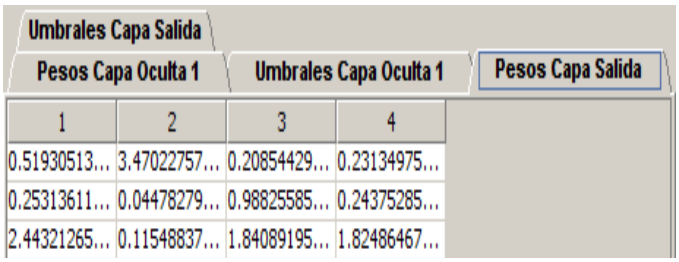

Fig. 6. RS (7,4) decoder ANN parameters

The description is done with VHDL code defining the set of adaptive network components. To which defines the conditions of the weight depending on the features enabled, as their respective background and consistent output.

To implement the network in the circuit described by VHDL language, there are various tools [15,16], artificial neural network, multilayer perceptron RNA (MPL, Multi-Layer Perceptrons), in [17] present a set of models.

For the configuration of network elements in VHDL was used Xilinx ISE 11 tool, that synthesized directly on the FPGA. Below is the code for hardware programming, with the architectural of MLP Neural Network, scheduled network defines the nodes of the input layer, hidden layer and output layer, with the respective synaptic weights, establishing the network architecture, with symbols of three bits in length, as specified in Table 1.

Table 1. RNA in VHDL

```
entity ANN_Decoder is
Port ( D7,D6,D5,D4,D3,D2,D1: in std_logic_vector(2 downto 0);
-- entradas al decodificador
    DECO3,DECO2,DECO1:out std_logic_vector(2 downto 0));
-- salidas del decodificador
end ANN_Decoder;
…
architecture Behavioral of ANN_Decoder is
…
Begin

--Input Layer
u1: neuron  port map (d7,w11,b11,ns11 );
u2: neuron  port map (d6,w12,b12,ns12 );
u3: neuron  port map (d5,w13,b13,ns13 );
u4: neuron  port map (d4,w14,b14,ns14 );
u5: neuron  port map (d3,w15,b15,ns15 );
u6: neuron  port map (d2,w16,b16,ns16 );
u7: neuron  port map (d1,w17,b17,ns17 );

-- Hidden Layer
uh1: neuron_sigmoid port map (ns11, ns12,ns13,ns14,ns15,ns16,ns17,wh11,wh12,wh13,wh14,wh15,wh16,wh17,bh1,ni1);
uh2: neuron_sigmoid port map (ns11, ns12,ns13,ns14,ns15,ns16,ns17,wh21,wh22,wh23,wh24,wh25,wh26,wh27,bh2,ni2);
uh3: neuron_sigmoid port map (ns11, ns12,ns13,ns14,ns15,ns16,ns17,wh31,wh32,wh33,wh34,wh35,wh36,wh37,bh3,ni3);
uh4: neuron_sigmoid port map (ns11, ns12,ns13,ns14,ns15,ns16,ns17,wh41,wh42,wh43,wh44,wh45,wh46,wh47,bh4,ni4);
--Output Layer
us1: neuron_identity port map (ni1,ni2,ni3,ni4,ws11,ws12,ws13,ws14,bs1,deco1);
us2: neurona_identity port map (ni1,ni2,ni3,ni4,ws21,ws22,ws23,ws24,bs2,deco2);
us3: neurona_identity port map (ni1,ni2,ni3,ni4,ws31,ws32,ws33,ws34,bs3,deco3);
end Behavioral;
```

There is parallel processing of inputs, due to the hardware configuration under *component* structures, which have concurrent processing and the relationship of the synapses of neurons associated with weight *w* that is configured according to training The description of the behavior of neuron component is based on the mathematical model, as show in equation 1.

$$\sum x_i w_{ij} = x_1 w_{11} + x_2 w_{11} + x_3 w_{11} + \ldots + x_{15} w_{11} \quad (1)$$

As the subscript *i* the identifier of the input, and the subscript *j* the identifier of the neuron in the layer. Table 2 presents the description of the behavior of the component neurons.

Table 2. Neurona in VHDL

```
entity neuron is
port    ( D: in std_logic_vector (2 downto 0);
w: in std_logic_vector (4 downto 0);
b: inout std_logic_vector (7 downto 0);
ni:out std_logic_vector (2 downto 0)  );
end neuron;

begin
bias <=b;
u1: multiplier      port map (D,w,prod);
u2: adders          port map (prod, bias, salida);
ni <= output (7 downto 5);
end Behavioral;
```

Similarly, has described each of the neurons and their components, to complete the structure of the ANN, obtained as a result the scheme for RTL network, as shown in Figure 7.

ISBN: 978-960-474-257-8

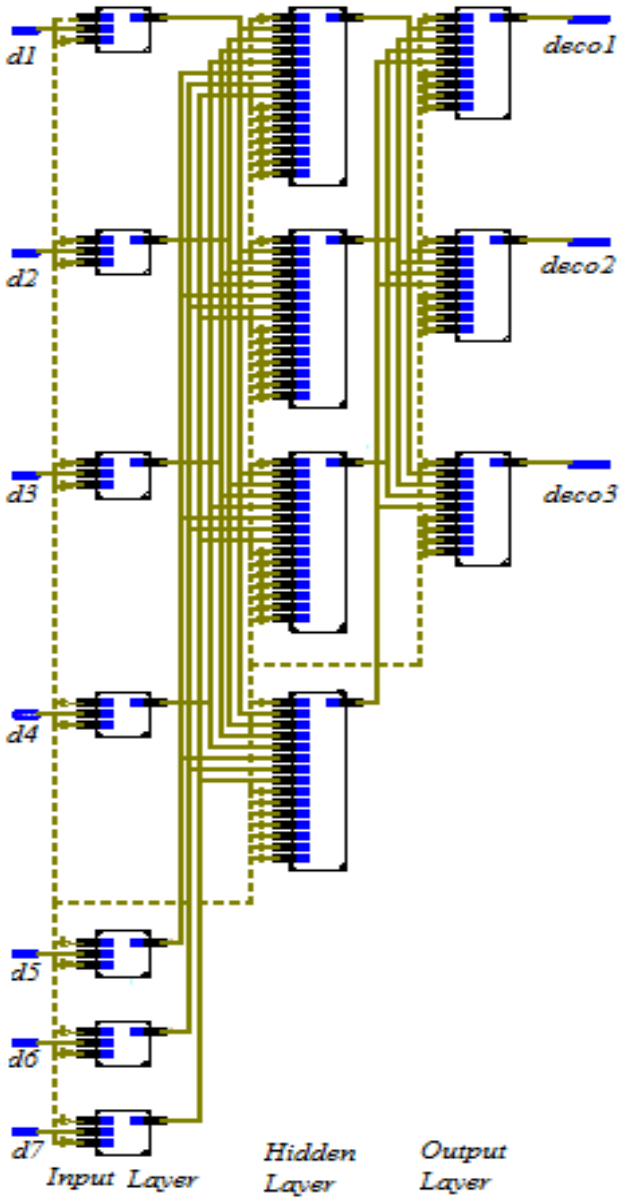

Fig. 7. RTL Schematic of RS(n,k) decoder on ANN

Of particular importance is the hardware implementation of the neuron activation function. The sigmoid function, traditionally used in ANNs, is not suitable for direct digital implementation as it consists of an infinite exponential series [16-17]. 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, with some examples of this technique reported in [2, 18]. A coarse approximation of the sigmoid function is the threshold (hard-limiting) function, as used in [19, 20]. The hidden layer in the design present neuron activation function of type sigmoid, this have been described through the approximation by ranks with base in the research of [21], as show in ec. 2.

$$f(x) = \begin{cases} 0 & si \quad x < -2 \\ 0.1572 & si \quad -2 \le x < -1 \\ 0.3340 & si \quad -1 \le x < 0 \\ 1/2 & si \quad x = 0 \\ 0.5734 & si \quad 0 < x \le 0.8 \\ 0.7356 & si \quad 0.8 < x < 1.5 \\ 0.8779 & si \quad 1.5 \le x \le 2.5 \\ 0.9509 & si \quad 2.5 < x \le 3.5 \\ 1 & si \quad 3.5 < x \end{cases} \quad \text{(ec. 2)}$$

The description of the neuron and *fun_act_sigmoide* component is presented in table 3.

Table 3. Behavioral of sigmoide neural

```
begin
bias <=b;
u1: multiplier port map (ns11,ws1,prod1);
u2: multiplier port map (ns12,ws2,prod2);
u3: multiplier port map (ns13,ws3,prod3);
u4: multiplier  port map (ns14,ws4,prod4);
u5: multiplier port map (ns15,ws5,prod5);
u6: multiplier  port map (ns16,ws6,prod6);
u7: multiplier port map (ns17,ws7,prod7);
u8: adder port map (prod1,prod2,prod3,prod4,prod5,prod6,prod7, bias, salida);
u9: fun_act_sigm  port map (salida, nsal);
nns<= nsal (7 downto 5);
end Behavioral;
-------------------------------------------------------------------------
architecture Behavioral of fun_act_sigm is
begin
process (ent)
begin
if ent < "10000010" then
nsal <="00000000";
elsif ent < "10000010" or ent > "10000001" then
nsal <="00011110";
--....
end if;
end process;
end Behavioral;
```

4 Conclusion

This investigation has achieved the goal of design, describe and synthesize hardware of a MLP neural network for the purpose of supporting an RS decoder (7,4), leaving the basis of the components of the network to generalize any other decoder, the MLP network results have been satisfactory, achieving an error in testing the RNA of 0.22, was able to obtain weights and bias parameters of reference with which to set up the decoder if selected studio. The synthesis report resulted in a use of slice 4%, 4 inputs lut tables

ISBN: 978-960-474-257-8

4% and of IOBs 19%, being the device used xc3s200-4pq208.

The module of the RNA has been described based on the architecture of the artificial neural network theory, the behavior was described by considering the topology of the network and the activation function of neurons in each layer, the system response depends on the training of the network from the data in the decoding module, the hardware training corresponds to the second phase of the project, in which from the input parameters and desired values are calculated the weights by the gradient descent algorithm property that the decoder based on artificial neural networks to adapt its parameters to real data in training time, thus offering an optimized output decoding time.

In comparison to previously done studies [5-7] the decoder presents contributions such as being a parallel decoder and fault-tolerant by the technology.

For the future, to create an interface allowing the software configuration of the network by setting the number of neurons in the input layer and output automatically, thus creating a neural network on reconfigurable hardware, where the parameters are weights and thresholds self-adjusting phase of training on the hardware given the training data.

References:

[1] Lakhmi, J., Advances in Evolutionary Computing for System Design, Springer, Springer, 2008

[2] 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

[3] Salsic, Z. (2006), "FPGA Prototyping of RNN Decoder for Convolutional Codes", EURASIP Journal on Applied Signal Processing Volume 2006, Article ID 15640, Pages 1–9

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

[5] Sandoval, C. y Fedón, A. "Codificador y decodificador Reed-Solomon programados a través de hardware reconfigurable", *Revista Ingeniería y Universidad*, Vol. 11, N. 1, Junio 2007, pp. 17-31

[6] Sandoval, C. y Fedón, A. "Programación VHDL de algoritmos de codificación para dispositivos de Hardware Reconfigurable", Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería, Vol. 24 (1), 2008, pp. 3-11.

[7] Sandoval, C. y Fedón, A. "Descripción modular de un esquema de codificación concatenado para corrección de errores con programación de hardware". *Ingeniare. Rev. chil. ing*. 2008, vol.16, n.2 [citado 2009-06-26], pp. 310-317.

[8] D. Hammerstrom, "A VLSI architecture for high-performance, low-cost, on-chip learning," in *Proceedings of International Joint Conference on Neural Networks IJCNN, 1990*, vol. 2, pp. 537–544

[9] Castiñeira M., J., Farrell Patrick G., "Codificacion para el control de errores", Primera Edición, Editorial Wiley, England, 2006

[10] Agatep A., "Reed-Solomon Solutions with Spartan-II FPGA", *Xilinx*, WP110, Vol. 1, N.1, February 10, 2000

[11] Moon T., "Error Correction Coding: Mathematical, methods and algorithms", ed. Wiley, 2005

[12] S. Coric, I. Latinovic, and A. Pavasovic, "A neural network FPGA implementation," in *Proceedings of the 5th Seminar on Neural Network Applications in Electrical Engineering NEUREL*, 2000, pp. 117–120

[13] J. Bruck and M. Blaum, "Neural networks, error-correcting codes, and polynomials over the binary *n*-cube," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 976–987, 1989

[14] Isasi P, "Redes Neuronales Artificiales (un enfoque práctico), Pearson Editorial, Madrid, 2004

[15] I. B. Ciocoiu, "Analog decoding using a gradient-type neural network," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 1034–1038, 1996

[16] 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

[17] 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

[18] S. L. Bade and B. L. Hutchings, "FPGA-based stochastic neural networks-implementation," in *Proceedings of IEEE Workshop on FPGAs for Custom ComputingMachines*, pp. 189–198, Napa Valley, Calif, USA, April 1994

[19] D. Abramson, K. Smith, P. Logothetis, and D. Duke, "FPGA based implementation of a Hopfield neural network for solving constraint satisfaction problems," in *Proceedings of Euromicro Conference (EUROMICRO '98)*, vol. 2, pp. 688–693, Vesteras, Sweden, August 1998.

[20] P. Larsson, "Error correcting decoder implemented as a digital neural network with a new clocking scheme," in *Proceedings of the 36th Midwest Symposium on Circuits and Systems*, vol. 2, pp. 1193–1195, Detroit, Mich, USA, August 1993

[21] Briceño J, "*Implementación hardware en una FPGA de una red neuronal artificial para el reconocimiento de patrones de voz específicos*", 2008