Arquitectura eficiente y de área reducida de una red neuronal estática para aplicaciones de procesamiento de señales e imágenes

Pantí de la Rosa, D. J.[1], Granados Cervera, J. G.[1], Méndez Méndez, J. A.[1], Castillo Atoche, A.[2], Sandoval Curmina, V.[1], Cruz Jimenez, B.[2], Moreno Sabido, M.[1]

Fecha de recepción: 29 de junio de 2017 – Fecha de aprobación: 18 de septiembre de 2017

RESUMEN

En este trabajo se propone una arquitectura de red neuronal para el desarrollo de aplicaciones para el procesamiento digital de señales e imágenes, utilizando técnicas recursivas y de cómputo paralelo con el fin de obtener un balance favorable de área y de velocidad de procesamiento. El diseño propone una estructura con niveles de pipeline que reduce significativamente los recursos de hardware en un dispositivo arreglo de compuertas programable en el campo (FPGA). La implementación del hardware se realizó con el sintetizador Xilinx XST ISE Web-Pack, y se obtuvieron 72 Slice Flipflops, 45 Slices, y 23 Luts, con la posibilidad de implementar hasta 103 neuronas en un FPGA Xilix Spartan3EXC3S500E.

Palabras Clave: Red Neuronal, Procesamiento Digital de Señales, Cómputo paralelo, FPGA.

An efficient and low resource static neural network architecture for signal and image processing applications

ABSTRACT

This paper proposes a novel artificial neural network (ANN) architecture for the development of signal processing applications. The presented approach implements parallel computing and recursive techniques, in a pipelined structure that balance the speed and hardware resources in a Field Programmable Gate Array (FPGA). Experimental results demonstrated a significant tradeoff between speed and hardware resources used in FPGA devices for signal processing applications. Based on Xilinx XST synthetizer, each neuron occupy 72 Slice Flipflops, 45 Slices, and 23 LUTs, which represents the possibility to implement up to 103 neurons in a FPGA Xilinx Spartan3E XC3S500E.

Keywords: Neural Network, Digital Signal Processing, Parallel computing, FPGA.

---

[1] Studies Division, Instituto Tecnológico de Mérida, Mérida, México

[2] Facultad de Ingeniería-UADY, Av. Industrias No Contaminantes por Anillo Periférico Norte s/n Apdo. Postal 150. Cordemex, Mérida, Yucatán, México.

Autor de correspondencia: acastill@correo.uady.mx

Nota: Este artículo de investigación es parte de Ingeniería–Revista Académica de la Facultad de Ingeniería, Universidad Autónoma de Yucatán, Vol. 21, No. 2, 2017, ISSN: 2448-8364.

I. INTRODUCTION

Artificial neural networks (ANNs) are computational algorithms inspired by the biological neural network of human brain (Dayhoff, 1990). ANNs presents a series of characteristics of the brain; they can learn from experience by changing their behavior depending on the environment and abstract the main characteristics of a series of data that apparently do not present common aspects (Hilera and Martínez, 1995). The objective of artificial neural networks is to simulate the way the human brain processes information since it is completely different from the operation of a computer that processes information in a linear way, while a neural network can perform parallel processing since it is highly complex and non-linear.

ANN contains a highly parallel distribution information processor, composed of multiple simple processing units called neurons that are characterized by having a natural inclination to acquire knowledge through experience, have a very high plasticity and great adaptability. They possess a high fault tolerance and has a non-linear behavior, allowing them to process information from other non-linear phenomena (Kröse and Smagt, 1996). Compared with software, the FPGA-based ANN implementation can utilize parallelism to speedup processing time. However, the implementation of ANN on FPGA devices deals with the design of complex arithmetic operations that require a tradeoff between precision and implementation (area and time performance). For example, standard floating-point representations minimize quantization errors while requiring significant hardware resources, and fixed-point representation may require less hardware resources but add quantization errors.

In this study, a novel ANN architecture is implemented using parallel computing and recursive techniques for the development of signal processing applications. The presented approach employs the Very High Speed Integrated Circuits Hardware Description Language (VHDL) and is implemented on a Xilinx Spartan3E XC3S500E chip. For the Log-Sigmoid, Gaussian and Hyperbolic Tangent function implementations, a pipelined parallel design has been employed. Finally, hardware resource comparative analysis results have been presented.

II. ARTIFICIAL NEURAL NETWORK BACKGROUND

An artificial neuron is a processing unit with three functional elements as shown in Figure 1.

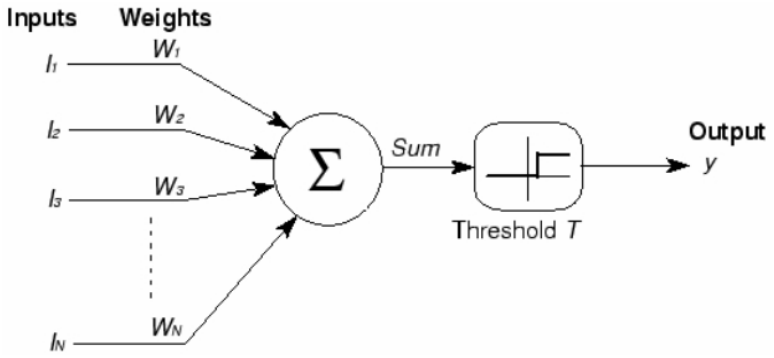


Figure 1. Design flow of an artificial neuron.

The basic elements of an ANN are: (1) a set of input nodes or pattern vector $\mathbf{l} = (l_1, l_2, \dots, l_N)^{\mathrm{T}}$, where one or more signals, usually from other neurons, are attenuated or amplified by a weight factor; (2) synaptic connections whose strengths are represented by a set of weights, here denoted by $\mathbf{w} = (w_1, w_2, \dots, w_N)^{\mathrm{T}}$; and (3) an activation

function ϑ that relates the total synaptic input to the output of the neuron. Here, ϑ is applied to the output of the adder to decide whether the neuron is activated.

The synaptic sum, u, to the neuron is given by the inner product of the input and weight vectors as follows:

$$u = \sum_{i=0}^{N} w_i l_i \quad (1)$$

where the threshold of the activation is applied to the synaptic input. The output of the neuron, y , is given by

$$y = \vartheta(u) \quad (2)$$

where $\vartheta(\cdot)$ is the neuron activation function.

Commonly used activation functions are:

- the Sigmoid activation function, where function is

$$\vartheta(u) = \frac{1}{1 + e^{-u}}, \quad (3)$$

- the hyperbolic tangent function

$$\vartheta(u) = \frac{2}{1 + e^{-2u}} - 1, \quad (4)$$

and the Gaussian function

$$\vartheta(u) = ce^{\frac{(u-b)^2}{2c^2}}, \quad (5)$$

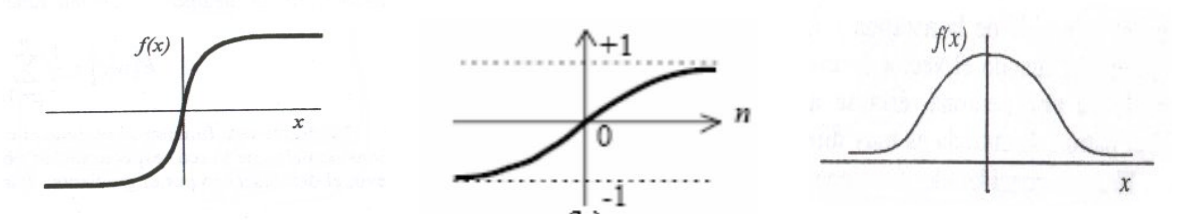
where b, c denotes real constants.

Figure 2 presents the graphic of each activation function considered in this study.

It is important to note here two critical points in relation to hardware implementation. First, parallelism can be justified only when high performance is attained at a reasonable hardware resource cost, and the Second, is that many techniques exist for implementing the neural network activation functions: polynomial approximations, the COordinate Rotation DIgital Computer (CORDIC) algorithm, rational approximations, table-driven methods, among others (Alimohammad, 2010). For hardware implementation, accuracy, performance and cost are all important, and many developed techniques in numerical analysis are not suitable for hardware implementation. For example, CORDIC is perhaps the most studied technique for hardware implementation, but it is rarely implemented: its advantage is that the same hardware can be used for several functions, but the resulting performance is usually rather poor.

III. HARDWARE IMPLEMENTATION

In this section, the hardware design implementation of the static neural network is presented. The architecture is conceptualized in two main blocks as illustrated in Figure 3: a finite impulse response (FIR) synapse and a pipelined non-linear activation function.



a) b) c)

Figure 2. Design flow of an ANN Commonly used activation functions: a) Sigmoid, b) Hyperbolic Tangent, c) Gaussian.

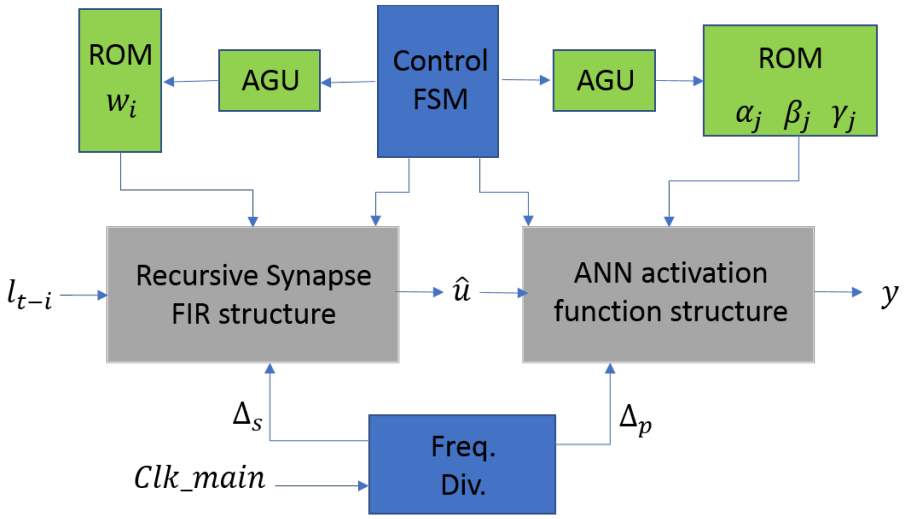


Figure 3. Static artificial neuron architecture.

The synaptic weight of (1) can be modeled as a FIR filter of digital filter theory. Using the method introduced by Lapedes and Farber (1987), the dependency of current outputs on previous inputs are modeled using a FIR synaptic model as follows:

$$\hat{u}(t) = \sum_{i=0}^{N} w_i \, l(t-i), \qquad (6)$$

where $\hat{u}(t)$is the synapse output at time t, and $l(t-i)$ is the delayed input to the synapse.

The architecture of the FIR synapse block is based on a signed 8-bit fixed-point Recursive Multiply-Accumulate Core (MAC) structure. As shown in Figure 4, the inputs of the neuron $l(t-i)$ are entered in a serial form, and the output $\hat{u}(t)$ is obtained after a defined number of clock cycles Δ_s.

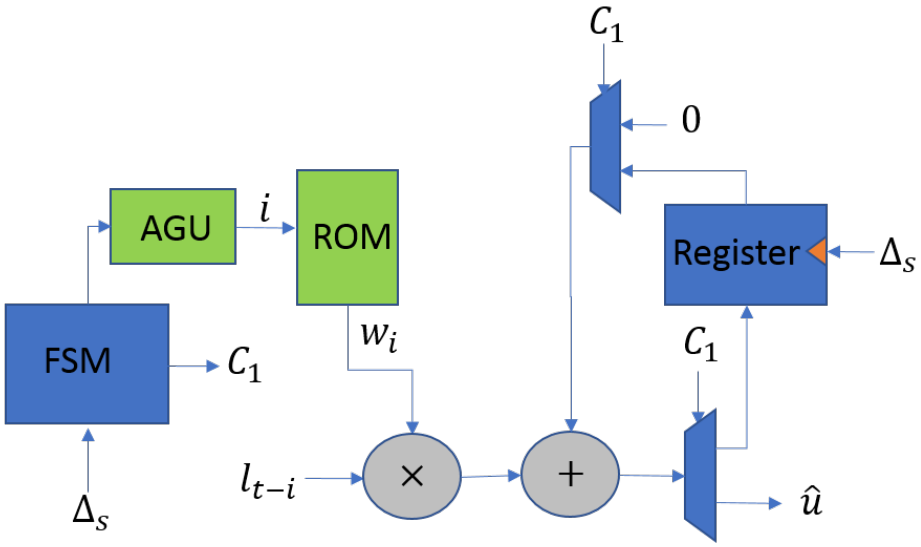


Figure 4. Recursive FIR synapse structure.

The FIR synapse block obtains its coefficients by pre-calculating and storing them in Read Only Memory (ROM) via an addressable generator unit (AGU) block. A control finite state machine (FSM) is the responsible to coordinate all operations. Due

to the serial nature of this block, the number of inputs determinate the total time of calculate the output processing time.

The hardware of the two-level pipelined ANN activation function is based on the Horner's Method and the series of Taylor approximation. The algorithm computes a third order polynomial, which approximates the Taylor series into a computationally efficient form. Given a polynomial $p(y) = \sum_{j=0}^{m} a_j y^j$, where $a_0, \ldots, a_m$ are real numbers, the approximation is represented as follows

$$p(y) = a_0 + y\Big(a_1 + y\big(a_2 + \cdots + y(a_{m-1} + a_m x)\big)\Big). \quad (7)$$

Figure 5 presents the fixed-point architecture of the pipelined ANN activation function.

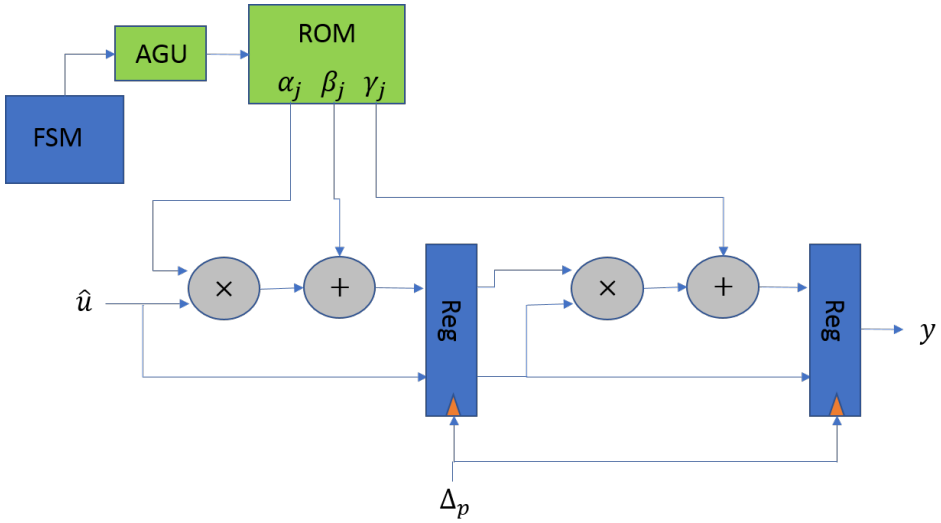


Figure 5: Pipelined ANN activation function architecture of the neuron.

In this study, the sigmoid, Gaussian and the hyperbolic tangent activation functions are developed and compared with other FPGA implementations as described in the next section.

IV. Experimental Results

The results of the area and time performance implementation of the static neural network architecture are presented in Table 1 and 2. In the area comparative analysis of Table 1, the Sigmoid, Gaussian and Hyperbolic Tangent activation functions were used considering the FPGA Spartan3E XC3S500E-4FG320 target device. The synthesis metrics specify the area and percent of utilization of the corresponding hardware cores. The HW architecture is designed using VHDL and synthesized using the Integrated Software Environment (ISE™) WebPACK™ 14.7 of the Xilinx XST tool.

Table 1. Device Utilization Summary for the static ANN

Device Utilization Summary			
ANN Activation Functions	Sigmoid	Gaussian	Hyperbolic Tangent
Number of Slice Flip Flops	72(1%)*	74(1%)	78(1%)
Number of 4 input LUTs	23(1%)	23(1%)	25(1%)

Number of occupied Slices	45(1%)	49(1%)	53(1%)
Number of bonded IOBs	26(11%)	26(11%)	26(11%)

*In parenthesis, percent of logic utilization based on FPGA Spartan3E XC3S500E-4FG320

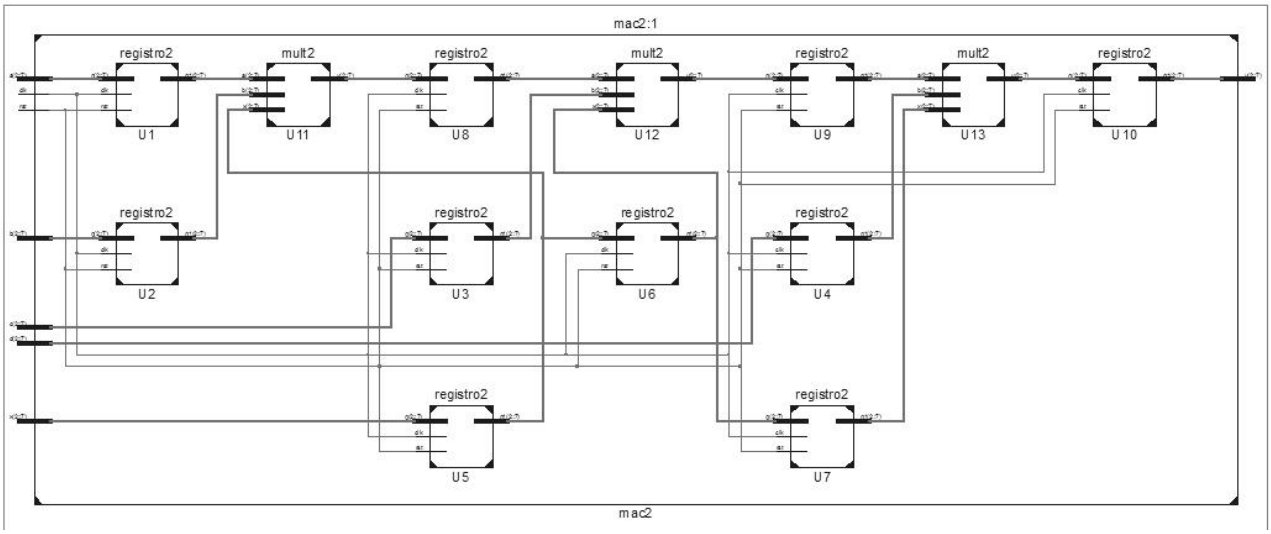


Figure 6. RTL Schematic of the MAC core using Horner's Method and three Pipeline levels.

Due the complexity of each activation function, the implementation has different levels of hardware resource utilization. The device utilization is directly proportional to the number of neurons that can be implemented in the FPGA, however, even in the most complex of the selected activation functions, it may be noted a small number of hardware resources without compromise the time performance implementation.

In this paradigm of design, it can be implemented multiple neurons working in an effective pipelined implementation as illustrated in Figure 6, being a cost-effective solution to different non-linear applications.
Based on area and timing reports of Xilinx XST, the maximum clock rate and neurons for this approach are shown in Table 2.

Table 2. Maximum Ratings for Implementation

Maximum Clock Rate	46 Mhz
Maximum Number of Neurons using Gaussian Function	94
Maximum Number of Neurons using Sigmoid Function	103
Maximum Number of Neurons using Hyperbolic Tangent Function	89

V. CONCLUSIONS

An static artificial neural network (ANN) architecture was designed for the development of signal processing applications. The design was implemented using parallel computing and recursive techniques, in a pipelined structure that balance the speed and hardware resources in a Xilinx Spartan3E XC3S500E FPGA. Particularly, an area and time comparative analysis of the Log-Sigmoid, Gaussian and Hyperbolic Tangent functions demonstrated a well-balance performance between speed and occupied hardware resources used in the FPGA.

REFERENCES

A. Alimohammad, S. F. Fard, B. F. Cockburn (2010), "A unified architecture for the accurate and high-throughput implementation of six key elementary functions," IEEE Trans. on Computers, 59(4), 449-456.

Dayhoff J. (1990). Early Adaptive Networks. Chapter 2 in "Neural Network Architectures. An introduction". Van Nostran Reinhold, New York, USA.

Fausett L. (1994). Simple Neural Nets for Pattern Classification. Chapter 2 in "Fundamentals of Neural Networks", 39-100, Prentice Hall, New Yersey, USA.

Haykin S. (1999). Information-Thoeretic Models. Chapter 10 in "Neural Networks". 506-565, Pearson Education, New Yersey, USA.

Hilera J.R. and Martínez V.J. (1995). Una Introducción a la Computación Neuronal. Chapter 1 In "Redes neuronales artificiales: Fundamentos, modelos y aplicaciones", 1-44 Ra-Ma Editorial, Madrid, Spain.

Knuth, D. E. (1998). Evaluation of Polynomials. Chapter 4.6.4 in "The Art of Computer Programming, Vol. 2: Seminumerical Algorithms", 466-506, Addison-Wesley, Massachusetts, USA.

Kröse, B. and Smagt, P. (1996). Fundamentals, Chapter I in "An introduction to neural networks", 11-20, University of Amsterdam, Amsterdam, Netherlands.

Este documento debe citarse como: Pantí de la Rosa, D. J., Granados Cervera, J. G., Méndez Méndez, J. A., Castillo Atoche, A., Sandoval Curmina, V., Cruz Jimenez, B., Moreno Sabido, M. (2017). **Arquitectura eficiente y de área reducida de una red neuronal estática para aplicaciones de procesamiento de señales e imágenes.** Ingeniería, Revista Académica de la FI-UADY, 21-2, pp. 23-29, ISSN 2448-8364.