Int. J. Electron. Commun. (AEÜ) 67 (2013) 513–520

Contents lists available at SciVerse ScienceDirect

International Journal of Electronics and Communications (AEÜ)

journal homepage: www.elsevier.com/locate/aeue

New pulse mode neuro-fuzzy hardware architecture and application to image denoising

Amir Gargouri*, Dorra Sellami Masmoudi

Computer Imaging and Electronics Systems Group at CEM Laboratory, University of Sfax, Sfax Engineering School, BP W, 3038 Sfax, Tunisia

ARTICLE INFO

Article history:
Received 30 April 2012
Accepted 28 November 2012

Keywords:
Pulse mode
Neuro-fuzzy system
FPGA implementation
Image denoising
On-chip learning

ABSTRACT

This paper focuses on a new digital architecture of pulse mode neuro-fuzzy system (PMNFS) with on-chip learning ability. The main purpose goal is to make use of the outstanding features of neuro-fuzzy in function approximation, and implement a reconfigurable architecture with on-chip learning on a field-programmable gate array (FPGA) platform. Details of the whole design with on-chip learning solutions are given. As an application illustrating the efficiency and scalability of the proposed PMNFS, we have considered the approximation of image denoising, which is a very important step in image processing. Experimental results show great efficiency of the proposed method, outperforming other denoising techniques. It was also demonstrated that such a system is strongly adaptive and gives good restored images independently of the kind of noises. Owing to learning, such feature cannot be met with conventional denoising techniques. Design synthesis results on a virtex II PRO FPGA platform are presented. Comparisons with conventional techniques as well as neural ones show higher performances of the designed PMNFS.

1. Introduction

It is well-known that both neural networks and fuzzy logic are intelligent techniques in information processing technology, which are frequently used for complex system modeling and data processing. Both techniques have distinctive features and present numerous advantages [1,2]. Fuzzy logic provides a framework to incorporate a knowledge base and to deal with imprecise data, whereas neural networks are endowed with various advantages such as parallelism, learning ability and speed of processing. However, Both techniques have their own limitations. Fuzzy logic is unable to design its fuzzy rules according to the training data to build a reconfigurable system able to identify the desired behavior, while the non-interpretability of results presents a major drawback of neural networks [3]. Owing to these problems, many researchers have combined neural networks and fuzzy logic synergistically as a single system, called hybrid neuro-fuzzy system (NFS), which brings both advantages from one part and overcomes the drawbacks from another part.

While, NFS has mostly been developed in software simulations, some other applications require restrictive design specifications such as high speed and reduced size for real-time operations [4,5]. In this context, VLSI implementation emerged as an adequate solution to integrate on-chip applications in artificial intelligence. Such an implementation allows real-time information processing, while using parallel architectures, as well as executing various applications in embedded systems [6]. However, hardware implementation of NFS with on-chip learning is an important and difficult issue, due either to the complexity of the membership functions, or to the overuse of the back-propagation algorithm, which requires a large number of multiplications and involves a great number of logical gates in hardware [7].

One of the efficient approaches to hardware implementation is the pulse mode use, where data are encoded as instantaneous and binary pulse signals. Indeed, pulse density systems afford a high-speed integration ability and present a new emerging technology to integrate various on-chip applications, thanks to its compact and simple solutions. Such mode provides a very simple multiplier architecture, which is an easier solution to overcome the limitation of the integration of conventional multipliers. To our knowledge, pulse mode architectures were only devoted to neural network implementation [8–12], and there has been no attempt to implement a hardware NFS based on pulse mode operations.

In this paper, we suggest an efficient approach to implement a reconfigurable pulse mode neuro-fuzzy system (PMNFS), able to perform all image processing steps as well as classification in a same application. The major concern is the approximation in image denoising functions, so that noise-free images could be processed in subsequent steps.

* Corresponding author: Tel.: +216 20851850; fax: +216 74848115.
E-mail addresses: gargouriamir@yahoo.fr (A. Gargouri), dorra.masmoudi@enis.rnu.tn (D.S. Masmoudi).

1434-8411/$ – see front matter © 2012 Elsevier GmbH. All rights reserved.
http://dx.doi.org/10.1016/j.aeue.2012.11.011

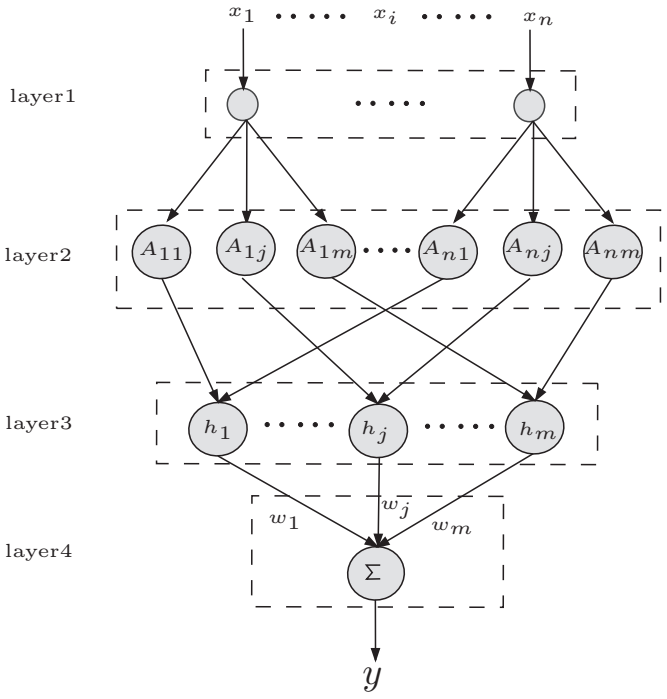

Fig. 1. Topological structure of the used NFS.

The remainder of this paper is organized as follows: Section 2 briefly overviews the NFS structure and introduces the particular model applied in this work. In Section 3, we present the hardware implementation of the PMNFS with on-chip learning ability. In Section 4, we describe the PMNFS design for image denoising. In Section 5, simulation results on a Virtex II-Pro platform (xc2vp7-6ff672) FPGA are reported and discussed, followed by a conclusion in Section 6.

2. Theoretical background

2.1. The structure of NFS

Several architectures using NFS approaches have been investigated in literature, such as ANFIS [13], DENFIS [14], FALCON [15], and FLEXNFIS [16]. In this work, in order to provide efficient hardware implementation and reduce the learning complexity, we are concerned with a modified version of Takagi–Sugeno inference system, requiring only four layers [17,18]. The topological structure of the used NFS model is shown in Fig. 1.

For a model with n inputs and one output, the fuzzy rule base can be expressed as the following form:

$$\textit{If } x_i \text{ is } A_{ij} \text{ and } \ldots \text{ and } x_n \text{ is } A_{nj} \text{ then } y \text{ is } w_j \tag{1}$$

where n is the number of input variables, m is the number of fuzzy rules, x_i is the input vector, A_{ij} is a fuzzy subset of x_i, y is the output variable, and w_j are the consequent parameters ($i = 1 \ldots n$ and $j = 1 \ldots m$).

Every layer has a role as follows:

Layer 1: no computation is performed in this layer. Input variables are simply transmitted without modification to the second layer.
Layer 2: nodes in this layer evaluate the membership degree of each input fuzzy set. Several membership functions can be applied as continuous or piecewise differentiable functions [13]. Here, the gaussian function is used. It is expressed by the following equation:

$$A_{ij}(x_i) = \exp - \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} \tag{2}$$

where c_{ij} and σ_{ij} are respectively centers and widths of membership functions, referred to as antecedent parameters.
Layer 3: nodes in this layer are fixed and referred to as rule nodes. They receive the output signals from the previous layer and aggregate the membership grades using fuzzy AND operations. Several T-norm operators performing fuzzy AND can be applied, such as minimum and product operations [2,6]. In this work, the min operator is applied; the output in this layer can be defined as follows:

$$h_j = min(A_{1j}, \ldots, A_{nj}) \tag{3}$$

Layer 4: this layer consists of a single node that gives the final inferred result. It computes the sum of all incoming signals generated by the previous layer, multiplied by the consequent parameters w_j. The output y of this node is calculated as:

$$y = \sum_{j=1}^{m} w_j h_j \tag{4}$$

2.2. Learning algorithm

This study uses the fully supervised gradient-descent method as a learning algorithm, it aims at minimizing some error criteria. The minimization is accomplished by adjusting the different parameters in an appropriate manner. The error function is often taken as the quadratic error and may be expressed as follows:

$$E = \frac{1}{2}(y - y_t)^2 \tag{5}$$

where y is the current output and y_t is the target.

According to the gradient-descent method, this error is closely linked to the different parameters and can be minimized by the following derivation:

$$\Delta\theta = -\rho_\theta \frac{\partial E}{\partial \theta} \tag{6}$$

where θ identifies the trained parameter and ρ_θ is a learning rate.

The consequent parameters w_j can be easily updated as follows:

$$\Delta w_j = -\rho_w (y - y_t) h_j(x) \tag{7}$$

$$w_j(t+1) = w_j(t) + \Delta w_j \tag{8}$$

In contrast to the consequent parameters, it is difficult to follow the derivative of the minimum function with respect to the antecedent parameters. As a consequence, we suggest to fix beforehand the values of c_{ij} and σ_{ij} and do not proceed in their update, using the functional definition suggested in [19], in which the membership function distributions are similar, so that they cover the full range of the input parameters.

3. Design and hardware implementation of PMNFS

Fig. 2 illustrates the block scheme of the PMNFS implemented architecture with on-chip learning ability. Mainly, the hardware design includes four principal units: a fuzzifier unit, an inference processing unit, a defuzzifier unit and a learning unit.

3.1. Fuzzifier unit

This module implements the fuzzification operator in Eq. (2), which uses the gaussian membership function. This work is drawn