UACEE International Journal of Advancements in Electronics and Electrical Engineering – IJAEEE
Volume 2 : Issue 3 [ISSN 2319 – 7498]
Publication Date : 09 September 2013

Digital ANFIS Architecture and Performance Analysis for Nonllinear Systems

Prasad R. Pande, Prashant L. Paikrao, Devendra S. Chaudhari

***Abstract*— Neuro Fuzzy systems, nowadays drawing attention of many researchers since being able to carry both neural networks and fuzzy logic's benefits. Adaptive Neuro Fuzzy Inference System (ANFIS) is such a neuro fuzzy architecture which has been widely accepted since invented. It is used in different applications like universal approximator, non-linear system realization, pattern recognition *etc*. However, because of variety of applications, implementation of ANFIS has been turn out to be specific and same implementation barely utilized with another one. FPGAs are potential enough to bring flexibility in hardware implementation of ANFIS so as to make it generic and application-independent. In this paper, an ANFIS model designing and implementation on FPGA is described. A unique dynamic ANFIS structure is realized with VHDL which is independent of system to be realized, membership function type used. It can be easily configurable with order of ANFIS, number of inputs/outputs, and number of membership functions. The study results illustrate that ANFIS presented in paper has been successfully realized with test nonlinear functions. Evaluations using standard error measurements revealed closer approximation of digital ANFIS to software one. The MATLAB simulation and FPGA implementation results clearly indicate that the hardware ANFIS model is acceptable in calibrate mode.**

***Keywords*—ANFIS, Digital System, FPGA, VHDL, Neuro Fuzzy System**

I. Introduction

Neuro-Fuzzy Systems (NFS) are centre of research and real world problems solutions from many recent years since they have advantages of both Artificial Neural Networks (ANN) and Fuzzy Inference Systems (FIS) [1]. Due to the growing need of adaptive intelligent systems to solve the real world problems NFS draws attention of many researchers which leads to different modelling techniques for either dedicated or generalized problems.

In recent years, hardware implementations of different neuro-fuzzy architectures like Adaptive Neuro-Fuzzy Inference System (ANFIS) have progressed a lot. Different neuro-fuzzy technologies emerged as optimal solutions for researchers in terms of speed of operation, cost, flexibility, usability and their trade-offs [2, 3]. Scientists have been provided with lot of flexibleness and versatility by neuro-fuzzy approaches so as to simulate or design their systems as per their requirements. It is researcher's and designer's responsibility to identify the perfect solution among available ones for their system by considering its feasibility and different trade-offs.

Real world systems demands for ANFIS hardware model rather than software with a tradeoff between versatility and performance [2, 4]. Since training process of ANFIS model cannot be driven with hardware because of memory dependency, it has to be carried out first as off-line in software. To take full advantage of such heterogeneous solution, integration of all parts of ANFIS is desirable so as to accommodate all the components of a typical embedded system on a single chip. Also the digital version of ANFIS should offer desired speed short time-to-market, re-usability, and availability of Intellectual Property (IP) cores with high flexibility [5].

Hence the intention behind development is to elaborate comprehensive architecture representing ANFIS model which can be widely used for different applications without using application dedicated resources. Furthermore, old model is aimed to be redesigned in a standard, extensively used hardware description language so as to be acceptable among designers and developers for FPGA implementations.

Amended digital architecture of ANFIS is presented here to realize on Field Programmable Gate Array (FPGA) using any hardware description language VHDL. Different three input nonlinear functions representing individual system are chosen for the analysis of first order ANFIS model. With respect to this function, ANFIS training was carried out in software and parameters set is obtained from it. After getting parameters list and membership values from training, digital architecture is modelled and implemented using hardware description language on FPGA. It used fixed point representation in language for positive integer inputs and signals to reduce model complexity. An attempt is made to design universal fuzzifier to reduce dependency on membership function type. ANFIS Architecture can be configured for necessary changes required for membership function numbers and order of inference polynomial outputs, if any. In the end of paper, moderate internal description of digital architecture of ANFIS is studied with performance and result analysis and paper concluded with further possible amendments respect to the model.

Prasad R. Pande
Government College of Engineering, Amravati
Maharashtra, India

Prashant L. Paikrao
Government College of Engineering, Amravati
Maharashtra, India

Dr. Devendra S. Chaudhari
Government College of Engineering, Amravati
Maharashtra, India

II. ANFIS

Adaptive Neuro-Fuzzy Inference System (ANFIS) was originally presented by Jang in 1993[6]. ANFIS is a hybrid neuro fuzzy technique that uses Fuzzy Logic to transform given inputs into a desired output through highly interconnected neural network processing elements and information connections, which are weighted to map the numerical inputs into an output. ANFIS combines the benefits of the two machine learning techniques (Fuzzy Logic and Neural Network) into a single technique. An ANFIS works by applying Neural Network learning methods to tune the parameters of a Fuzzy Inference System.

To present the ANFIS architecture, two fuzzy IF-THEN rules based on a first order Sugeno model are considered:

Rule (1): **IF** x is A_1 **AND** y is B_1, **THEN** $f_1 = p_1x + q_1y + r_1$.
Rule (2): **IF** x is A_2 **AND** y is B_2, **THEN** $f_2 = p_2x + q_2y + r_2$.

Where:
• x and y are the inputs,
• A_i and B_i are the fuzzy sets,
• f_i are the outputs within the fuzzy region specified by the fuzzy rule, and
• p_i, q_i and r_i are the consequent parameters that are determined during the training process.

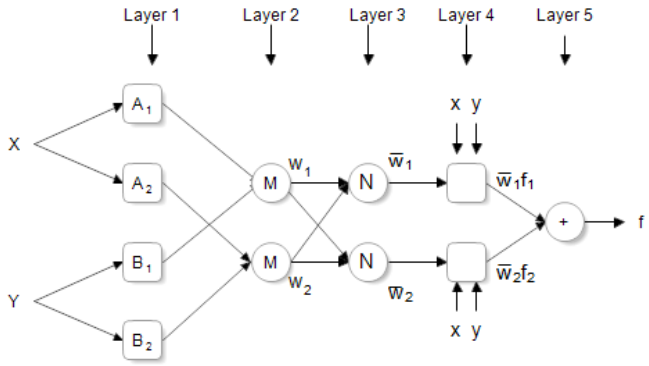

Figure 1 – Generalized ANFIS Architecture[6]

The ANFIS architecture employing Sugeno inference model to implement above two rules is shown in Figure 1. In this figure, a circle indicates a fixed node, whereas a square indicates an adaptive node. ANFIS has five-layer architecture. Each layer is explained in detail below.

In Layer (1), all the nodes are adaptive nodes. The outputs of Layer (1) are the fuzzy membership grade of the inputs, which are given by the following equations:

$$O_{1,i} = \mu_{A_i}(x), \quad \text{for i = 1, 2, or} \tag{1}$$

$$O_{1,i} = \mu_{B_{i-2}}(y), \quad \text{for i = 3, 4} \tag{2}$$

Where x and y are the inputs to node i, and A_i and B_i are the linguistic labels associated with this node function. $\mu_{A_i}(x)$ and $\mu_{B_{i-2}}(y)$ can adopt any fuzzy membership function. For example, if the bell shaped membership function is employed, $\mu_{A_i}(x)$ is given by:

$$\mu_{A_i}(x) = \frac{1}{1+\left|\frac{x-c_i}{a_i}\right|^{2b_i}} \tag{3}$$

Where a_i, b_i and c_i are the parameters of the membership function.

In Layer (2), the nodes are fixed nodes. This layer involves fuzzy operators; it uses the **AND** operator to fuzzify the inputs. They are labeled with M, indicating that they perform as a simple multiplier. The output of this layer can be represented as:

$$O_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y) \quad \text{for i = 1, 2} \tag{4}$$

These are the so-called firing strengths of the rules.

In Layer (3), the nodes are also fixed nodes labeled by N, to indicate that they play a normalization role to the firing strengths from the previous layer. The output of this layer can be represented as:

$$O_{3,i} = \bar{w_i} = \frac{w_i}{w_1+w_2}, \quad \text{for i = 1, 2} \tag{5}$$

Outputs of this layer are called normalized firing strengths.

In Layer (4), the nodes are adaptive. The output of each node in this layer is simply the product of the normalized firing strength and a first order polynomial (for a first order Sugeno model). The output of this layer is given by:

$$O_{4,i} = \bar{w_i} f_i = \bar{w_i}(d_i x + e_i y + g_i) \tag{6}$$

Where d_i, e_i, and g_i are the consequent parameters.

In Layer (5), there is only one single fixed node labeled with '+'. This node performs the summation of all incoming signals. The overall output of the model is given by:

$$O_{5,i} = \sum_i \bar{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{7}$$

The learning algorithm for ANFIS is a hybrid algorithm that is a combination of gradient descent and least squares methods. In the forward pass of the hybrid learning algorithm, node outputs go forward until Layer 4 and the consequent parameters are determined by the least-squares. In the backward pass, the error signals propagate backward and the premise parameters are updated using gradient descendent. The hybrid learning approach converges much faster by reducing search space dimensions of the original back propagation method [7, 8].

ANFIS model is not a structurally rigid adaptive network. Layer 3 and 4 can be merged together and overall ANFIS could result into four layer structure. In that case, weight normalization is performed at very last stage [8]. This theory has been into consideration while designing the presented digital ANFIS system.

III. System Development of Digital ANFIS

FPGA implementable architecture of ANFIS was pioneered by two researchers, H.J.B. Saldana and C.S Cardenas. They have first implemented it for two input nonlinear function [9]. Furthermore, architecture was modified for three input single output nonlinear function by them [10]. The system was described in hardware descriptive language and successfully implemented on FPGA.

In presented work, old architecture is ameliorated to bring flexibility for 3 input nonlinear systems. The training for selected nonlinear functions is done off-line in MATLAB environment. A script is developed which can be modified for any numbers and types of membership functions and epoch numbers. It is also calibrated to give positive integers for membership values and consequent parameters of driven ANFIS. Once training done, all these values and parameters are unchanged during further system operation and stored in memories of respective sub-blocks of digital system.

The digital architectural design of ANFIS is divided into four subsystems – Fuzzifier, Permutator, Inference and Defuzzifier as shown in Figure 2. In this, three numeric positive integer inputs are represented by *l*, *m*, and *n* which exemplify nonlinear system's inputs. The *output* is the overall ANFIS system output and *ready* signal indicate end of system operation or system readiness for another set of inputs. Other than these, there are some control signals like clock and reset which can be given from FPGA kit to be used for implementation. Brief narration about each subsystem is given below.

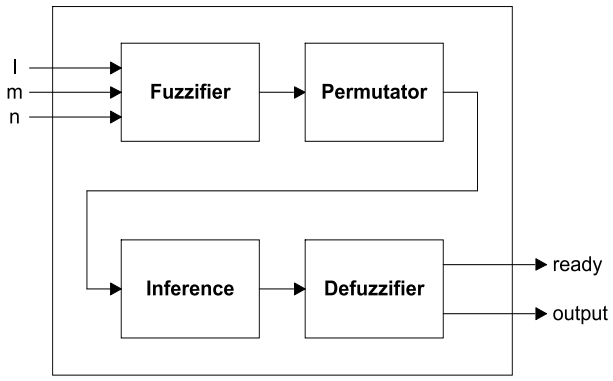

Figure 2 – Digital ANFIS Architecture

A. *Fuzzifier*

The first subsystem confers membership values for each of the three inputs *l, m* and *n*. As shown in Figure 3, it has total three -

- Read-only-memories (store_l, store_m, store_n) used to store the membership function values for each input. These are directly exported from MATLAB script dynamically.
- Compare and Mapping blocks (Compare_l, Compare_m and Compare_n) that, as named, compares and gives appropriate membership values of their respective input.

It also comprised a controller called Fuzz_controller to manage all the operations flow. All membership values are stored in respective ROMs first for initial clock cycles given to Fuzzifier. After that, every external input is compared and mapped with respective ROM membership values and at last given out through *l_Fout, m_Fout and n_Fout* signals. *fuzz_ready* signal is generated to indicate Permutator that values are available for further processing. A reset to this block clears all stored values and whole block operation starts from initial steps.

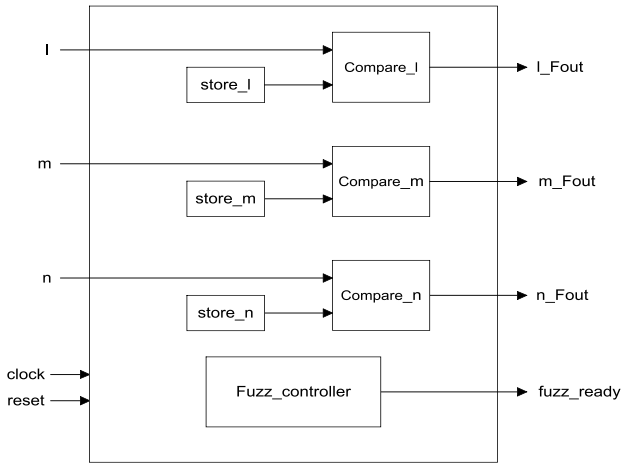

Figure 3 - Fuzzifier subsystem

B. *Permutator*

The Permutator stores membership values that are forwarded by the Fuzzifier subsystem. Each possible permutation of these values is generated so that all membership values can be multiplied by the next subsystem. This operation has to be done to get different rule strengths for inference operation.

Permutator comprised of three circular shift registers- Circ_fun_l, Circ_fun_m and Circ_fun_n, where the storage and permutation process is carried out, and a controller as shown in Figure 4. Operation of Permutator starts with signal *fuzz_ready* from Fuzzifier. Each set of rule strength (weight) factors are tracked with *seq_add* signal to keep synchronization with consequent parameters in next Inference subsystem. Signals *finish* and *per_ready* indicates termination of permutation process and operation of Permutator respectively. Internal control signals from controller are omitted to avoid complexity.

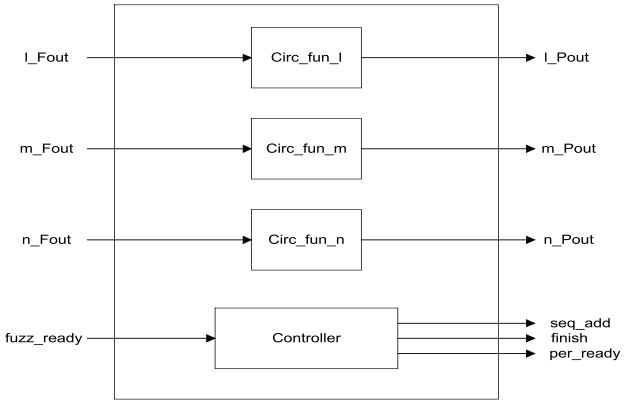

Figure 4 - Permutator subsystem

C. *Inference*

Inference subsystem is responsible for calculating rule firing strengths. It has following blocks and a controller shown in Figure 5.

- Poly_Gen has inputs from the user as well as from Seq_ROM which stores the consequent parameters obtained from script. It gives Sugeno inference outcomes in synchronization with *seq_add* signal.
- Two MAC (Multiplier-Accumulator) units consisting Multiplier, Storage registers and Accumulator each for calculation of Numerator and Denominator of equation (7). MAC unit1 comprises Multiplier 1, Reg_W and Den_Acc gives output *Den* while MAC unit2 – Multiplier 2, Reg_FW and Num_Acc collectively work and give output signals *Num*. In MAC units, set of membership values are multiplied with each other and with polynomial function to obtain the weights, firing strength of rules.
- Controller which controls the overall operation with respect to signals from previous subsystem and internal design of block.

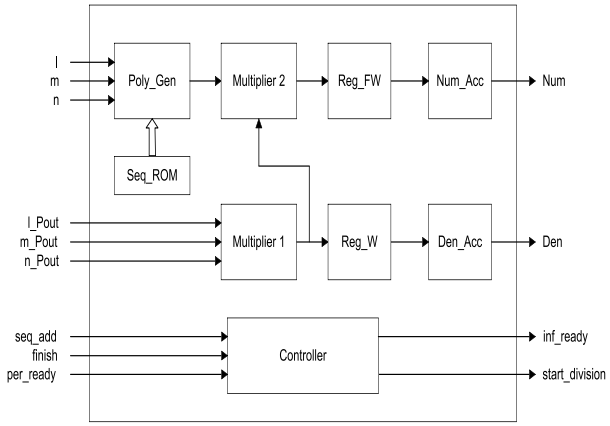

Figure 5 - Inference subsystem

D. *Defuzzifier*

The Defuzzifier subsystem carry out the division operation between signals *Num* and *Den* obtained from Inference subsystem. The result of this division *output* signifies to the output of the ANFIS. It contains a divider, and a controller as shown in Figure 6. Its function is described by the following equation:

$$output = \frac{Num}{Den} = \frac{\sum_i f_i w_i}{\sum_i w_i} \quad (8)$$

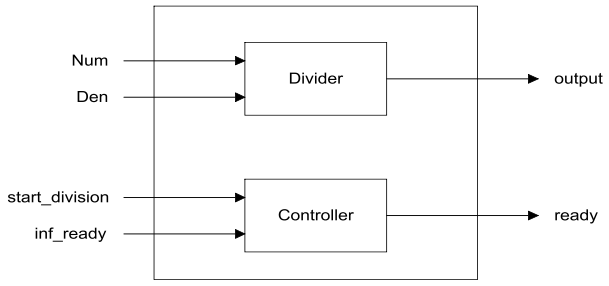

Figure 6 - Defuzzifier subsystem

IV. System Performance and Result Analysis

In this section, simulation and FPGA implementation results of ANFIS are discussed along with their performance regarding to nonlinear functions. These nonlinear functions represent nonlinear systems chosen for testing purpose. A script is written for individual functions and training is carried out in MATLAB 7.12 to get input membership values and consequent parameter set. After training, membership values and consequent parameters are converted into binary fixed point numbers to be stored in respective ROMs inside ANFIS design. ANFIS architecture is then designed and simulated using ALDEC Active-HDL version 9.1 (Expert Edition). Later, for FPGA implementation, ALTERA DE2 Kit is used which has FPGA - Cyclone II device family. Arbitrary binary input values are presented to the system and their results are verified with MATLAB results obtained for respective function.

Following are 4 test nonlinear functions chosen all with *gbellmf*, epoch no. -20 and 3 membership functions per input which can be configurable.

$$f_1(l,m,n) = (1 + l^{0.5} + m^{-1} + n^{-1.5})^2, \quad \text{l,m,n} \in [1,6] \quad (9)$$

$$f_2(l,m,n) = 100 * l * e^{(l^{0.1} - m^{0.2} - n^{0.04})}, \quad \text{l,m,n} \in [1,4] \quad (10)$$

$$f_3(l,m,n) = 38 * (\sin(l^{0.4}) * m^{0.5} + \log 2^k), \quad \text{l,m,n} \in [1,5] \quad (11)$$

$$f_4(l,m,n) = \frac{l^m + m^n + n^l}{17 * l^{(0.2m)}}, \quad \text{l, m} \in [1,5] \ \& \ \text{n} \in [1,4] \quad (12)$$

Figure 7 illustrates results of MATLAB as well as FPGA for 1st nonlinear function in equation (9). As it is clearly seen that Output lines are almost overlapped indicating the accuracy of the system.

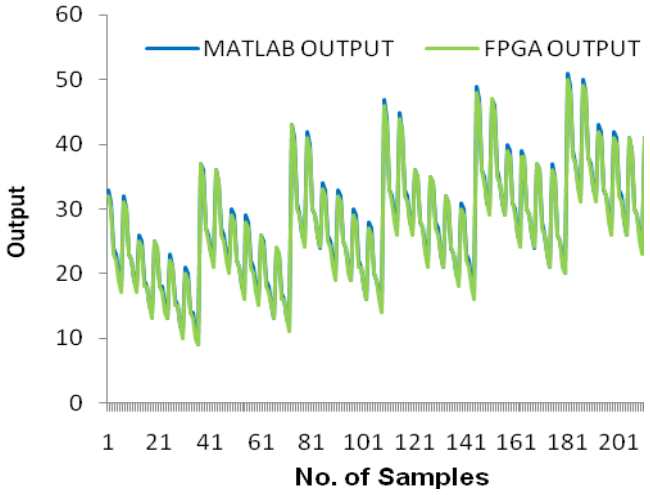

Figure 7 – Graph between MATLAB and FPGA results for equation (9)

The preciseness of developed model can be determined from percentage error evaluated for FPGA outcomes and MATLAB results below in Figure 8 for same function.

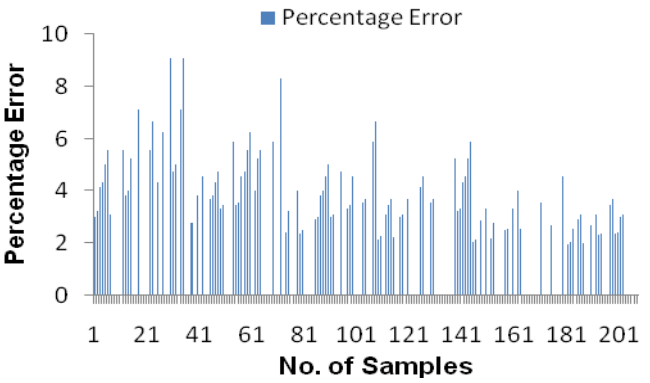

Figure 8 – Percentage Error Graph for all samples tested for equation (9)

In order to measure correctness and performance of nonlinear systems chosen for experiment, Mean Square Error (MSE) is calculated which is briefed in below table I.

TABLE I. MEAN SQARE ERROR OF NONLINEAR FUNCTIONS

Nonlinear Function	No. of Samples	Mean Square Error (MSE)
Function 1	216	0.5509
Function 2	64	0.6875
Function 3	125	0.5120
Function 4	100	0.5300

v. Conclusion with Future Extents

Early ANFIS architecture has design complexity with membership function type and approach is developed with zero order which could be impractical in real world. Also, it has been observed that the performance varies significantly with nonlinear function change.

Presented modified architecture ensures universal fuzzifier calculations since it removes dependency on type of membership function. It also works with any order and membership function numbers with little redesigning. MATLAB simulation and FPGA implementation results shows that there is acceptable error level obtained and model can be widely used for different nonlinear systems in calibrated mode.

Presented work focuses on reducing the design and bringing dynamic nature in it rather than using knotty floating point calculation. The architecture of ANFIS could be amended with negative integer input range as well in future.

References

[1] Nauck D., "Neuro Fuzzy Systems: Review and Prospects", *European Congress Intelligent Techniques and Soft Computing (EUFIT'97)*, pp. 1044–1053, 1997

[2] Echanobe J., Campo I. D., Bosque G., "An adaptive neuro-fuzzy system for efficient implementations", *International Journals on Information Sciences*, pp. 2150–2162, 2008.

[3] Aldair A., Wang W., "FPGA Based Adaptive Neuro Fuzzy Inference Controller For Full Vehicle Nonlinear Active Suspension Systems", *International Journal of Artificial Intelligence & Applications (IJAIA)*, Vol.1, No.4, pp 1-15, 2010.

[4] Campo I. D., Echanobe J., Bosque G., Tarela J. M., "Efficient Hardware/Software Implementation of an Adaptive Neuro-Fuzzy System" , *IEEE Transactions on Fuzzy Systems*, Vol. 16, No. 3, pp.761-778, 2008.

[5] Sulaiman N., Obaid Z. A., Marhaban M. H. and Hamidon M. N., " FPGA- Based Fuzzy Logic: Design and Applications – a Review", *IACSIT International Journal of Engineering and Technology*, Vol.1, No. 5, pp 491-503, 2009.

[6] Jang J. S. R., "ANFIS: Adaptive-network-based fuzzy inference systems", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp 665-685, 1993.

[7] Jang J. S. R., Sun C.T., "Neuro-fuzzy modeling and control", *Proceedings of the IEEE*, Vol. 83, No.3, pp 378-406, 1995.

[8] Jang J. S. R., Sun C.T., Mizutani E., *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, Upper Saddle River NJ, USA, 335-368, 1997.

[9] Saldana H.J.B., Cardenas C.S., "Design and implementation of an adaptive neuro fuzzy inference system on an FPGA used for nonlinear function generation", *IEEE ANDESCON*, pp. 1-5, 2010.

[10] Saldana H.J.B., Cardenas C.S., "A digital architecture for a three input one output zero-order ANFIS", *IEEE Third Latin American Symposium on Circuits and Systems (LASCAS)*, pp. 1-4, 2012.