Fuzzy Flip-Flop Based Neural Networks as a Novel Implementation Possibility of Multilayer Perceptrons

Rita Lovassy
Inst. of Microelectronics and Technology
Kandó Kálmán Faculty of Electrical Engineering
Óbuda University Budapest, Hungary
lovassy.rita@kvk.uni-obuda.hu

László Gál
Dept. of Technology, Informatics and Economy
University of West Hungary Szombathely
gallaci@ttmk.nyme.hu

Árpád Tóth
Dept. of Automation, Faculty of Engineering Sciences
Széchenyi István University Győr, Hungary
toth.zen@gmail.com

László T. Kóczy
Dept. of Informatics, Faculty of Engineering Sciences,
Széchenyi István University Győr, Hungary
Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics,
Budapest, Hungary
koczy@sze.hu

Imre J. Rudas
Inst. of Intelligent Engineering Systems, John von Neumann
Faculty of Informatics
Óbuda University, Budapest, Hungary
rudas@uni-obuda.hu

Abstract—Fuzzy Flip-Flop based Neural Networks (FNN) constructed from fuzzy D flip-flops are studied as a novel technique to implement multilayer perceptrons. The starting point of this approach is the concept of fuzzy flip-flop (F3), as the extension of the binary counterpart. Fuzzy D flip-flop based neurons are viewed, as sigmoid function generators. Their characteristic equations contain simple fuzzy operations, thus enabling easy implementability. FNNs have an interconnected fuzzy neuron structure composed from a large number of neurons acting in parallel which are capable of learning, and are suitable for function approximation. In this paper we propose the FPGA implementation of Łukasiewicz operations, furthermore of fuzzy D flip-flop neurons based on Łukasiewicz norms.

Keywords - function approximation, fuzzy flip-flop, fuzzy neural network, hardware realization of Łukasiewicz type fuzzy flip-flop neurons

I. INTRODUCTION

The multilayer perceptron (MLP) is a widely used artificial neural network type. The MLP as function approximator is the object of study in different engineering applications as well as applied mathematics and computer science.

In [5] we introduced Fuzzy Flip-Flop based Neural Networks (FNN), a multilayer perceptron based on fuzzy flip-flops. The proposed network is a structure consisting of the same type of fuzzy flip-flops. In the FNN the neurons have been substituted by fuzzy J-K flip-flops with feedback (in reality, a type of combinational circuit) and fuzzy D flip-flops derived from the fuzzy J-K F3 based on frequently used fuzzy operations (e.g. algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade norms).

In our previous papers [5-7] several types of fuzzy J-K and D flip-flop based on various fuzzy operations and standard complementation have been proposed. We determined their characteristic equations, illustrating and comparing their properties. It was shown that broadly they may be classified into two groups, one presenting quasi s-shape transfer characteristics and the rest having non-sigmoid character. The construction of a fuzzy neuron unit from fuzzy J-K and D flip-flops was proposed indicating a possible connection of this fuzzy unit with the artificial neuron, the basic component of neural networks.

The main idea is to have such a fuzzy neural network structure which can be implemented simply in hardware. The fuzzy norms, furthermore simple fuzzy flip-flop characteristic equations are not complicated in hardware realization. We propose the FNN as a neural network with general-purpose hardware which could be more user-friendly; it is not bound to algorithmic a-priori-assumptions and therefore offers high flexibility. Neurohardware might provide a much better cost-to-performance ratio, lower power consumption and smaller size [4].

In [10] we compared the function approximation capability of FNNs with one and two hidden layers based on Dombi, Łukasiewicz and new Trigonometric (1) norms. During the simulations for simple comparison an equal number of neurons in the hidden layers were chosen, increasing their number step by step. We concluded that FNNs are more suitable to avoid overfitting than standard neural networks (e.g. the tansig function based MATLAB Neural Network Toolbox networks) in the frame of simple function approximation problems, the experimental results show that these FNNs provide rather good
generalization performance, with far better mathematical stability than the tansig based neural networks, and they are more suitable to avoid overfitting in the case of test data containing noisy items in the form of outliers. Comparing the simulation results, we concluded, that Łukasiewicz type fuzzy D flip-flop neurons based NNs with two hidden layers performed best; they had the best generalization capability.

This paper is organized as follows. After the Introduction, in Section II the sigmoid function generator derived from fuzzy D flip-flops based on Łukasiewicz operations are briefly reviewed. The 8 bit FPGA implementation of Łukasiewicz t-norm and t-conorm is presented in Section III. Section IV shows furthermore the internal structure of a Łukasiewicz type fuzzy D flip-flop based neuron from two approaches: first from the functional analysis point of view, and second with respect to the timing hazards. The Fuzzy D Flip-Flop Based Neural Network architecture is presented in Section V. The suggested neural network has two hidden layers having fuzzy flip-flop neurons (as sigmoid function generators) and a single linear output node. Finally, some concluding remarks are given.

II. SIGMOID FUNCTION GENERATORS DERIVED FROM FUZZY D FLIP-FLOP

A. Fuzzy D Flip-Flop Based on Łukasiewicz Norms

The Łukasiewicz t-norm is defined as

\[ i_{\text{Luka}} = \max \{0, x + y - 1\} \] (1)

and t-conorm is [3]

\[ u_{\text{Luka}} = \min \{x + y, 1\} \] (2)

The characteristic equation of fuzzy D flip-flop is given by [5]

\[ Q(t+1) = (D u_{\text{Luka}} D) i_{\text{Luka}} (D u_{\text{Luka}} Q) i_{\text{Luka}} (D u_{\text{Luka}} (1-Q)) \] (3)

Substituting the Łukasiewicz norms into equation (3) the D F³ characteristic equation can be defined as

\[ Q(t+1) = \max \{0, -1 + \max \{0, R\} + \min \{1, S\}\} \]

\[ R = -1 + \min \{1, 2D\} + \min \{1, D + Q\} \] (4)

\[ S = 1 + D - Q \]

Based on equation (4) in our previous paper [8] we proposed the construction of a fuzzy D flip-flop neuron which is a combinational sigmoid generator. Figure 1 shows the sigmoidal transfer characteristics shape for a fixed, optimized [6] internal state \( Q = 0.18 \).

III. FPGA IMPLEMENTATION OF ŁUKASIEWICZ NORMS

A. Digital Equivalent of Łukasiewicz Norms

This subsection starts with the digital representation of Łukasiewicz norms. We propose integer numbers in the interval \([0, 2^N-1]\), instead of floating point numbers whose truth value can be infinite between \([0, 1]\), where \(N = 8\) is the number of the bits used, and \(2^{255} = 255\) is the complete membership function [9]. By using \(x, y\) as integers, if \(I = 2^N - 1\) is the maximum value [11] the new forms of Łukasiewicz norms are

\[ i_{\text{Luka}} = \max \{0, x + y - I\} \] (5)

\[ u_{\text{Luka}} = \min \{x + y, I\} \] (6)

The binary subtraction process in equation (5) can be substituted by an addition operation by summing the negated value of the second operand [2]. In other words, instead of performing the subtraction \(x + y - I\) the addition operation \(x + y + (-I)\) is made. Recall that in two’s complement representation, to negate a value involves inverting all 0’s to 1’s and 1’s to 0’s, and then adding a 1. Once the value of \(I = 2^8 - 1\) (11111111, the 8 bit equivalent) is applied, instead of the 11111111 form we use the 00000000 + 1 = 00000001 representation. Thus, the Łukasiewicz t-norm expression can be rewritten as

\[ i_{\text{Luka}} = \max \{0, x + y + 1\} \] (7)

B. 8 Bit FPGA Implementation of Łukasiewicz T-Norm

Figure 2 shows the 8 bit FPGA implementation of Łukasiewicz t-norm according to equation (7). The combinational circuit consists of an 8 bit full adder (ADD8B) that implements the \(x + y\) part. The full adder carry input (CI) is set to a voltage level corresponding to integer 255 to realize the \(x + y + 1\) operation. The second part of the circuit is an 8 bit 2-to-1 multiplexer (MUX). A multiplexer is needed in order to select which one of the inputs (DA, DB) will be the output of the circuit. The selection is made according to the full adder carry output (CO) value, which fulfils the maximum operation.

If the sum result for \(x + y + 1\) value is greater than the 8 bit representation maximum value (255), it sets a carry output signal (CO = 1), and the result is driven as final result. If CO signal from the adder is not set (CO = 0) it is necessary to bound the final result to 0 by connecting the MUX DA input to integer 0 (ground) level.
Figure 2. Architecture of implemented Łukasiewicz t-norm

C. 8 Bit FPGA Implementation of Łukasiewicz T-Conorm

Figure 3 shows the 8 bit FPGA implementation of Łukasiewicz t-conorm according to equation (6). In a similar way, as in subsection B, the combination circuit is built up in part from an 8 bit full adder (ADD8B). The CI input is set to a constant 0 value, to realize the $x + y$ operation, in addition the carry out (CO) output controls the 8 bit 2-to-1 multiplexer (MUX) Select line. The MUX task is to select the minimum value between the sum $x + y$ and the $I$ value. In this approach if the addition result for input values is greater than $I$ (out of the interval) the output must be bounded by $I$. In this case the carry output value will be 1 (CO = 1). This value is set by connecting the DB input to integer 255 (11111111). If the CO signal from adder is not set (CO = 0), it means that the addition result is smaller than $I$, the multiplexer shows the minimum value of the final output, which in this case is equal to the sum $x + y$.

Figure 3. Architecture of implemented Łukasiewicz t-conorm

IV. FPGA IMPLEMENTATION OF THE ŁUKASIEWICZ TYPE FUZZY D FLIP-FLOP NEURON

A. 8 Bit FPGA Implementation of Łukasiewicz Type Fuzzy D Flip-Flop Neuron

According to equation (3), the combinational circuit which describes the fuzzy flip-flop neuron (see Figure 4) is composed from three blocks belonging to the Łukasiewicz t-norm, two units which stay instead of the Łukasiewicz t-conorm, and one single inverter.

The input of the circuit is driven by a synchronized clock pulse in the sample-and-hold circuit which could be a simple D flip-flop used as register for holding the D input, denoted by U3/FD8CEB.

For a fix internal state $Q = 0.18$ [6] the simulation results (made by Xilinx Spartan-3AN, xc3s 1400) show the sigmoid transfer characteristics (see Figure 5) which presents the same behavior as we obtained it in [7] as a result of calculations (Figure 1).

B. Timing Hazards

In this subsection we focus on timing issues of fuzzy D flip-flop based neuron built up from Łukasiewicz norms. Figure 4 presents such a neuron architecture where it is assumed that there is no delay. In this ideal case, we had not considered the timing of the circuit. The simulation results (see Figure 5) show the fuzzy D flip-flop neuron behavior when the circuit is studied only with respect to its fuzzy operation, i.e. a functional analysis is taken.

When a circuit is actually implemented, the timing of the circuit must be considered with care, taking into account the time for the signals to pass from the input of a fuzzy gate to the output. We made the timing analysis of fuzzy D flip-flop neuron according to the architecture introduced in Figure 4, and the results are concluded in the next timing diagram (see Figure 6).

This simulation captured accurately the presence of the timing hazard. Synchronized to the falling edge of the clock signal the output of the circuit reached a constant value unfortunately, after a glitch.

In the next we will propose an alternative circuit, derived from the original one (Figure 4) by introducing a new unit. The task of the new 8 bit register is to eliminate the glitch caused by multiple sources having paths of different delays driving that signal. The new model registers are synchronized. The offset between the two register clock pulse (CLK_1PH and CLK_2PH) was selected in a way to eliminate the propagation delay. The quarter of a one clock period should be greater than the maximum of the appeared hazard interval. In our simulations we tune the clock pulses phase delay to $\pi / 2$.

As indicated in Figure 7, which is to compare to Figure 4 this architecture is completed with the second register, denoted by U13/FD8CEB.

Simulation results are presented in Figure 8 where the value of $Q$ is fixed to 0.18 (in our digital representation corresponding to 46).
Figure 4. Architecture of FPGA implemented Łukasiewicz type fuzzy D flip-flop neuron

Figure 5. Łukasiewicz type fuzzy D flip-flop neuron sigmoid transfer characteristics obtained from simulations

Figure 6. Łukasiewicz type fuzzy D flip-flop neuron timing diagram based on the architecture presented in Figure 4
C. Fuzzy D Flip-Flop Based Neural Network (FNN)

The proposed FNN (Figure 8) is a four-layered Multilayer Perceptron with hidden nodes defined by fuzzy D flip-flop neurons (as sigmoid function generators) and a linear output node. This network presented rather good function approximation properties [7, 8].

In our approach the weighted input values are connected to input D of the new fuzzy flip-flop neuron based on a pair of Łukasiewicz type t-norm and t-conorm, having quasi sigmoid transfer characteristics. The output signal is then computed as the weighted sum of the input signals transformed by the transfer function.

Our investigations included comparative experiments using the (standard) tansig based functions for the simulated neurons that we always used as reference [7, 8]. The most suitable type of $F^3$ neurons for constructing fuzzy neural networks was the Łukasiewicz type fuzzy D flip-flop neuron.

As our major motivation in these investigations was to construct a technology for the creation of real hardware we found that MLPs and $F^3$ based FNNs obviously offered much simpler and cheaper possibility for hardware implementation compared to a rather complicated approximated tansig type neural network.
V. CONCLUSIONS

Observing the behavior of FPGA implementation of the FNN that was presented in this paper it can be stated that this approach provides a safe and hazardless solution to the construction of an easily hardware implementable neural network that is suitable to be deployed in an environment where observation might be distorted by a component of noise. On the other hand the hardware architecture presented in this paper is far for being the simplest and cheapest. It is definitely possible to simplify the hardware construction while not affecting any functional feature and maintaining the protection against hazards by applying certain easily identifiable steps of minimization of the circuit. In the future this simplified architecture will be determined and new FPGA implementations corresponding to the simplified circuitry will be studied. We expect that results comparable to the once presented here will be achieved by applying the new circuits. Other FNNs based on different fuzzy operators and F3 types will be similarly investigated.

ACKNOWLEDGMENTS

This work is partially supported by Óbuda University Grants, the project TAMOP 421B, a Széchenyi István University Main Research Direction Grant, further National Scientific Research Fund Grant OTKA K75711.