Hardware Implementation of Fuzzy Flip-Flops Based on Łukasiewicz Norms

RITA LOVASSY1,2, ANTONIO HERNÁNDEZ ZAVALA3, LÁSZLÓ GÁL1,4, OSCAR CAMACHO NIETO3, LÁSZLÓ T. KÓCZY1,6, ILDAR BATYRSHIN7

1 Inst. of Informatics, Electrical and Mechanical Engineering Faculty of Engineering Sciences
Széchenyi István University Győr, HUNGARY
2 Inst. of Microelectronics and Technology, Kandó Kálmán Faculty of Electrical Engineering
Óbuda University Budapest, HUNGARY
3 Dept. of Electronics, Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas
Instituto Politécnico Nacional, MÉXICO
4 Dept. of Technology, Informatics and Economy
University of West Hungary, HUNGARY
5 Dept. of Microelectronics and Embedded Systems, Centro de investigación en Computación
Instituto Politécnico Nacional, MÉXICO
6 Dept. of Telecommunication and Media Informatics
Budapest University of Technology and Economics, HUNGARY
7 Instituto Mexicano del Petroleo, MÉXICO

lovassy.rita@kvk.uni-obuda.hu, antonioh_z@hotmail.com, gallaci@ttmk.nyme.hu, oscarc@cic.ipn.mx, koczy@sze.hu, koczy@tmit.bme.hu, batyr1@gmail.com

Abstract: - The digital hardware implementation of various fuzzy operations furthermore of fuzzy flip-flops has been the subject of intense study and application. The fuzzy D flip-flop derived from fuzzy JhK one is a single input - single output unit with sigmoid transfer characteristics in some particular cases, proper to use as neuron in a Fuzzy Neural Networks (FNN). In this paper we propose the hardware realization of fuzzy D flip-flops based on Łukasiewicz norms.

Key-Words: - Fuzzy Flip-Flop, Fuzzy Neuron, Fuzzy Neural Networks, Łukasiewicz Norms, Łukasiewicz type Fuzzy Flip-Flop, hardware realization

1 Introduction

There are several different manners to combine fuzzy set theory, artificial neural networks and bacterial evolutionary algorithms, the three main branches of Computational Intelligence (CI), which may differ essentially according to the approaches and the tasks. Many applications show an increased interest in using either two or all three of them in one system. Fuzzy systems are transparent and interpretable, neural networks possess the property of auto-adaptability and the bacterial algorithms have optimal structure approximation properties.

In our previous papers [4], [5], [6] several types of fuzzy flip-flop have been proposed in a manner that covers possible combinations of fuzzy J-K and fuzzy D flip-flops based on various fuzzy operations, determining their basic definitions and properties, including also graphical illustrations, comparisons. It was shown that broadly they may be classified into two groups, one presenting quasi s-shape transfer characteristics and the rest with non-sigmoid character. The construction of a single input-single output unit from fuzzy J-K and D flip-flops was proposed. This fact indicates a possible connection of this fuzzy unit with the artificial neuron the basic component of neural networks.

In [5] we introduced the Fuzzy Flip-Flop based Neural Networks (FNN). The proposed network is a structure consisting of the same types of F3 whose training is much simpler than training the individual flip-flops. In the FNN the neurons have been substituted with fuzzy J-K flip-flop with feedback and fuzzy D flip-flop derived from the fuzzy J-K one based on frequently used fuzzy operations (e.g. algebraic, Łukasiewicz, Yager, Dombi, Hamacher, Frank and Dubois-Prade norms). The main idea is using the property of learning from the data of the network using a parallel structure which can be implemented in hardware. It is known that simple parametrical t-norms, furthermore simple fuzzy flip-flop characteristic equation are uncomplicated for tuning and hardware realization. Rudas et al. [10] proposed the hardware implementation of a generation of parametric families of fuzzy connectives together with min-max, Łukasiewicz and drastic t-norms and co-norms. In [12] Zavala et al. used FPGA technology to implement the above mentioned norms into an 8 bit single circuit that allows operation selection. From these basic operations,
more complex structures can be constructed to be adapted to a wider variety of fuzzy applications. To accomplish this, blocks of basic t-norm and t-conorm operations are proposed as add-in modules for existing fuzzy hardware. These blocks can be interconnected to obtain different configurations to choose the appropriated configuration for a specific purpose, increasing versatility of fuzzy hardware. In this paper we propose the hardware implementation of fuzzy D flip-flop based on Łukasiewicz fuzzy operations. The paper is organized as follows. After the Introduction Section 2 deals with fuzzy J-K and D flip-flops in general, furthermore with Łukasiewicz operations based fuzzy D flip-flops. The FNN properties and the Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM) used for neural networks parameter optimization and training is carried out in Section 3. Section 4 presents the details of the hardware realization of the fuzzy D flip-flops based on Łukasiewicz operations (LF³). Finally, in Section 5 a brief Conclusion and References.

2 Fuzzy Flip-Flops (F³)

The fuzzy J-K flip-flop is an extended form of the binary J-K flip-flop [2]. In this approach the truth table for the J-K flip-flop is fuzzified, where the binary AND, OR and NOT operations are substituted by their respective fuzzy counterparts, i.e. t-norm, t-conorm and fuzzy negation respectively.

The next state \(Q_{t+1}\) of a J-K flip-flop is characterized as a function of both the present state \(Q\) and the two present inputs \(J\) and \(K\). The so called fundamental equation of the J-K type fuzzy flip-flop [2], [9] is

\[
Q_{t+1} = i_1(u(D, K), i_1(u(J, Q), u(K, Q)))
\]  

(1)

Where \(i\) and \(u\) denote fuzzy operations, in particular fuzzy conjunction, disjunction and negation respectively (e.g. \(\bar{K} = 1 - K\)). As a matter of course, it is possible to substitute the standard operations by any other reasonable fuzzy operation triplet (e.g. De-Morgan triplet), thus obtaining a multitude of various fuzzy flip-flop pairs. The fuzzy flip-flop was proposed as the basic unit in fuzzy register circuits [9]. In [6] the construction of a single input-single output unit from fuzzy J-K flip-flops where \(Q\) is fed back to \(K (K = 1 - Q)\) and (old) \(Q\) is fixed is proposed. Connecting the inputs of the fuzzy J-K flip-flop in a particular way, namely, by applying an inverter in the connection of the input \(J\) to \(K\), case of \(K = 1 - J\), a fuzzy D flip-flop is obtained. Substitute \(K = J\) in equation (1) and let \(D = J\), the fundamental equation of fuzzy D flip-flop is

\[
Q_{t+1} = i(u(D, D), i(u(D, Q), u(D, \bar{Q})))
\]  

(2)

We have conducted extensive investigations by simulations and found that the \(J \rightarrow \bar{Q}(t+1)\) transfer characteristics of fuzzy J-K flip-flops based on Łukasiewicz, Dombi, Yager, Hamacher, Frank and Dubois-Prade norms, further the \(D \rightarrow \bar{Q}(t+1)\) characteristics of (new) fuzzy D flip-flops of Łukasiewicz, Yager, Hamacher, Frank and Dubois-Prade operations show quasi sigmoidal curvature, while all other F³’s investigated have non-sigmoidal behavior.

2.1 Łukasiewicz Operations Based Fuzzy D Flip-Flops (LF³)

The Łukasiewicz norms are

\[
i_1(x, y) = \max(0, x + y - 1)
\]  

(3)

\[
u_1(x, y) = \min(x + y, 1)
\]  

(4)

The fundamental equation of the Łukasiewicz type fuzzy D flip-flop [9] can be rewritten in the form

\[
Q_{t+1} = i_1(u_1(D, D), i_1(u_1(D, Q), u_1(D, \bar{Q})))
\]  

(5)

3 Fuzzy Flip-Flops Based Neural Networks (FNN)

In general, two trainable layer networks with sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer are universal approximators [3]. The neuro-fuzzy system proposed is based on two hidden layers constituted from fuzzy flip-flop neurons. The FNN is a supervised feedforward network, applied in order to approximate test functions. The nonlinear characteristics exhibited by fuzzy neurons are represented by quasi sigmoid transfer functions. The proposed network activation function is the same at each hidden layer, from unit to unit. The function approximation goodness is strongly dependent on the number of fuzzy neurons in the hidden layers [11].

3.1 Parameter Optimization

In [1],[8] we proposed the BMAM which is a particular combination of evolutionary and gradient based algorithm for training fuzzy flip-flop based neural networks to optimize the network variable values to improve function approximation performance. The core of BMAM contains the bacterial mutation, and its basic idea is to improve the parts of chromosomes contained in each bacterium. The bacterial mutation mechanism is
used by the bacteria which can transfer genes to other bacteria. To find the optimal approximation for our network we encoded our FNN weights, biases, \( Q \) and fuzzy operation parameter values in a bacterium. Therefore a procedure is working on changing the variables, testing the model obtained in this way and selecting the best models.

In [7] we found the optimal \( Q \) and fuzzy operation parameter pair for J-K and D type \( F \)'s based on algebraic, Yager, Dombi and Hamacher norms by training a 1-8-8-1 FNN with the Bacterial Memetic Algorithm. Using the same method we identified the optimal value of \( Q = 0.18 \) at the end of the BMAM training the network consist of Łukasiewicz type fuzzy D flip-flops.

By extensive simulations [4] we found that the fuzzy J-K flip-flops based on Dombi, furthermore the fuzzy D flip-flops based on Łukasiewicz, Yager and Hamacher norms are the most suitable ones for constructing FNNs. As these FNN types produced more or less low MSE values in all simulation experiments.

4 Hardware Realization of Łukasiewicz Fuzzy D Flip-Flops

4.1 Digital Considerations for t-norms and t-conorms

In order to construct efficient digital hardware for Łukasiewicz based fuzzy D flip-flops, some considerations must be analyzed respecting to truth space values and operation definitions.

4.1.1 Truth Space

Truth space values for fuzzy numbers are commonly expressed as floating point numbers whose truth value can be infinite between [0, 1], where 0 means non-membership and 1 means complete membership. Unfortunately it is not so easy for a computer to do calculations using floating point representation. In order to reduce hardware resources consumption and increase computing speed, we can work with integers over a universe \( U = [0, 2^m - 1] \), where \( m \) is the number of bits used to represent truth space and defines resolution, then 0 corresponds to the no-membership value, and \( 2^m - 1 \) is the complete membership value (255 when \( m = 8 \)).

4.1.2 Operations Redefinitions

Taking into account these considerations, equations (3) and (4) from section 2.1 can be rewritten as (6) and (7) using \( x, y \) as integers, and let \( I = 2^m - 1 \) be the maximum value [11].

\[
\begin{align*}
i_L(x, y) &= \max\{0, x + y - I\} \\
u_x(x, y) &= \min\{x + y, I\}
\end{align*}
\]

Łukasiewicz operations based fuzzy D flip-flop is constructed using two basic hardware blocks corresponding to operations presented on (6) and (7), these blocks are presented first separately, then final hardware for Łukasiewicz type fuzzy D flip-flop is shown along with its simulations and hardware resources consumed.

4.2 Digital Hardware for Bounded Product Operation, Łukasiewicz t-norm

Circuit presented in Fig. 1, corresponds to bounded product operation presented by equation (6) where \( I \) is the maximum value over truth space, with 8-bit resolution, according to considerations from Section 4.1. First block from left \( \text{(ADD8)} \), is an 8 bit adder to implement \( x + y \) part, if the sum result for this values is greater than 8 bit maximum value (255), it sets a carry output signal \( (CO=1) \) and is used to indicate that \( I \) can be subtracted from adder result on middle block \( \text{(ADSU8)} \) that is an 8 bit subtractor. If \( CO \) signal from adder is not set \( (CO=0) \), it means that addition result is smaller than (or equal to) \( I \) so then when \( I \) is subtracted, result will be negative (or 0) and is necessary to bound final result to 0. Final block on right side \( \text{(MUX_2_8B)} \) is a two eight bit input multiplexer driven by \( SEL \) signal that is connected directly to \( CO \) from adder, if this signal is equal to 0 it lets final output be bounded to 0, otherwise, result from subtractor \( x + y - I \) is driven as final result.

Simulation for bounded product is presented on Fig. 2. for different input values, on left side labels for each input are presented for \( I, X, Y \) and output result \( BP \), all of them are 8 bit wide. It can be seen at output results that when addition of input values is less than \( I \), result is bounded to 0, otherwise valid results are shown.

4.3 Digital Hardware for Bounded Sum Operation, Łukasiewicz t-conorm

Circuit presented in Fig. 3, corresponds to bounded sum operation presented by equation (7) where \( I \) is the maximum value over truth space, with 8-bit resolution, according to considerations from Section 4.1. First block from left \( \text{(ADD8)} \), is an 8 bit adder to implement \( x + y \) part, if addition result for input values is greater than 8 bit maximum value (255), it sets carry output signal \( (CO=1) \) indicating that is out of bounds and must be bounded to \( I \). If \( CO \) signal from adder is not set \( (CO=0) \), it means that addition result is smaller than \( I \) then it is driven as final result on right side block \( \text{(MUX_2_8B)} \) that is a two eight bit input multiplexer driven by \( SEL \) signal that is...
connected directly to $CO$ from adder, if this signal is equal to 0 it lets final output be the result of adder, otherwise, result is bounded to $I$.

Simulation for bounded sum is presented on Fig. 4, for different input values, on left side labels for each input are presented for $I$, $X$, $Y$ and output result $BO\_SUM$, all of them are 8 bit wide. It can be seen at output results that when addition of input values is greater than $I$, result is bounded to $I=255$, otherwise valid results are shown.

Fig. 1. Hardware realization for bounded product or Łukasiewicz t-norm. First adder $ADD8$ realizes $x + y$, then subtractor $ADSU8$ completes $x + y - I$. Carry output from adder ($CO$) helps to decide which output is lead as output on final multiplexer ($MUX\_2\_8B$), if it is not activated, result is bounded to 0.

Fig. 2. Simulation results for bounded product or Łukasiewicz t-norm. It can be seen at output results ($BP$) that when addition of input values is less than $I$, result is bounded to 0, otherwise valid results are shown.

Fig. 3. Hardware realization for bounded sum or Łukasiewicz t-conorm. It consists of an adder ($ADD8$) and a two eight bit input multiplexer ($MUX\_2\_8B$) driven by $SEL$ signal that is connected directly to $CO$ from adder, if this signal is equal to 0 it lets final output be the result of adder, otherwise, result is bounded to $I$.

Fig. 4. Simulation results for bounded sum or Łukasiewicz t-conorm. It can be seen at output results ($BO\_SUM$) that when addition of input values is greater than $I$, result is bounded to $I=255$, otherwise valid results are shown.
Fig. 5. Hardware realization for Łukasiewicz type fuzzy D flip-flop. It is composed of three BOUNDEDSUM blocks, two BOUNDEDPROD blocks one 8 bit register and an inverter.

Fig. 6. Simulation results for Łukasiewicz type fuzzy D flip-flop. It can be seen that it presents sigmoid characteristics.

Table 1. Hardware resources consumed by Łukasiewicz operations and overall fuzzy D flip-flop.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Logic Gates</th>
<th>Logic Levels</th>
<th>I/O used</th>
<th>Total Delay ns</th>
<th>Delay Logic ns</th>
<th>Delay Route ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounded Product</td>
<td>134</td>
<td>17</td>
<td>32</td>
<td>13.87</td>
<td>10.46</td>
<td>3.4</td>
</tr>
<tr>
<td>Bounded Sum</td>
<td>91</td>
<td>15</td>
<td>32</td>
<td>13.75</td>
<td>8.04</td>
<td>5.71</td>
</tr>
<tr>
<td>LFFF</td>
<td>432</td>
<td>42</td>
<td>35</td>
<td>26.55</td>
<td>17.17</td>
<td>9.37</td>
</tr>
</tbody>
</table>

4.4 Fuzzy D Flip-Flop Based on Łukasiewicz Operations

Final implementation for Łukasiewicz type fuzzy D flip-flop is presented in circuit on Fig. 5. It is constructed using blocks presented on sections 4.2 and 4.3 as BOUNDEDPROD and BOUNDEDSUM respectively, satisfying equation (5). FD8RE is an 8 bit simple D flip-flop used as register for holding D input. Inputs for the circuit are D, Q, I and CLK, other inputs are fixed values. Output is expressed as Qtx1 and corresponds to Q_{i+1} where we can identify the sigmoid characteristics. To obtain Q just one 8 bit inverter was required.

Simulation results are presented on Fig. 6, where the value of Q is fixed to 0.18 that in our digital representation corresponds to 46. This value was obtained as optimal value according to section 3. On left side there are labels for inputs I, Q, D and CLK, RESET and CLK_ENABLE are fixed values to let D register work.

Hardware resources consumed are presented on Table 1 for each Łukasiewicz operations as on circuits presented on Fig. 1. and Fig. 3. and for the overall Łukasiewicz based fuzzy D flip-flop. Number of logical gates consumed, logic levels required, I/O used and timing delay are expressed in nanoseconds.
5 Conclusion

FPGA technology was used to implement Łukasiewicz type fuzzy D flip-flops. The proposed circuit was built up using core blocks which realize Łukasiewicz t-norm and t-conorm. The implementation is an 8 bit data width architecture and additionally two bit as selection for operations. These blocks can also be chained to form more complex structures such as FNN. The hardware resources consumed by Łukasiewicz operations and overall fuzzy D flip-flops are presented. In the future we intend to realize hardware implementation of more complex structures which allow parametric or trigonometric operations.

Acknowledgements

This work was partially supported by the National Scientific Research Fund Grant OTKA K75711, further by Széchenyi István University Main Research Direction Grant 2010, Budapest Tech Grants and Bilateral CONACYT-NKTH project No. 10110/127/08 Mod. Ord. 38/08, SIP-IPN project 20100847. NKTH.

References:


