Modeling and Simulation of the Defuzzification Stage of a Type-2 Fuzzy Controller Using the Xilinx System Generator and Simulink

Roberto Sepúlveda[1], Oscar Montiel[1], Gabriel Lizárraga[2], and Oscar Castillo[3]

[1] Centro de Investigación y Desarrollo de Tecnología Digital del Instituto Politécnico Nacional, Av. del Parque No.1310, Mesa de Otay, 22510, Tijuana, B.C., México
r.sepulveda@ieee.org,o.montiel@ieee.org
[2] M.S. Student at CITEDI-IPN
lizarraga@citedi.mx
[3] Division of Graduate Studies and Research,
Calzada Tecnológico S/N, Tijuana, B.C., México
ocastillo@hafsamx.org

Abstract. This paper is focused on the study, analysis and development of code for the defuzzification stage of type-2 fuzzy systems, through the average of two type-1 fuzzy systems. This proposal is based on the average method for systems where the type-2 membership functions of the inputs and output, have no uncertainty in the mean or center. The codification is done using the hardware description language VHDL, and it was exported to Simulink through the Xilinx System Generator (XSG). Comparative tests were conducted between the type-2 fuzzy systems for different number of bits and noise levels.

1 Introduction

Most of the existing implementations of type-1 or type-2 fuzzy systems have been achieved in software on general-purpose computers, the main drawback of these implementations is the speed limitation due to the sequential computer program execution [26]. Thus the other option for the implementation of fuzzy systems that require high processing speed to operate in real-time is based on dedicated hardware. In this line of development, in order to achieve high performance systems, the use of VLSI devices is increasing [21]. The FPGA is a VLSI device highly flexible that allows us to implement digital circuitry through the use of specialized software, which is an appealing characteristic for designers. Other good features are that they consume low power and can be reprogrammables in field. Hence, there is an increasing interest in using FPGA devices to design digital controllers, and a growing interest in control systems based on fuzzy logic [13, 27, 29]. In fact it is possible to find some works where type-1 fuzzy inference systems (FIS)have been implemented in FPGA, such as in [26] for electrical vehicles, a Mamdani FIS based on FPGA using distributed arithmetic to calculate the defuzzification stage [17], or diverse designs and implementations on FPGA of fuzzy inference systems [1, 16, 22, 23, 24, 30].

O. Castillo et al. (Eds.): Evolutionary Design of Intelligent Systems, SCI 257, pp. 309–325.
springerlink.com

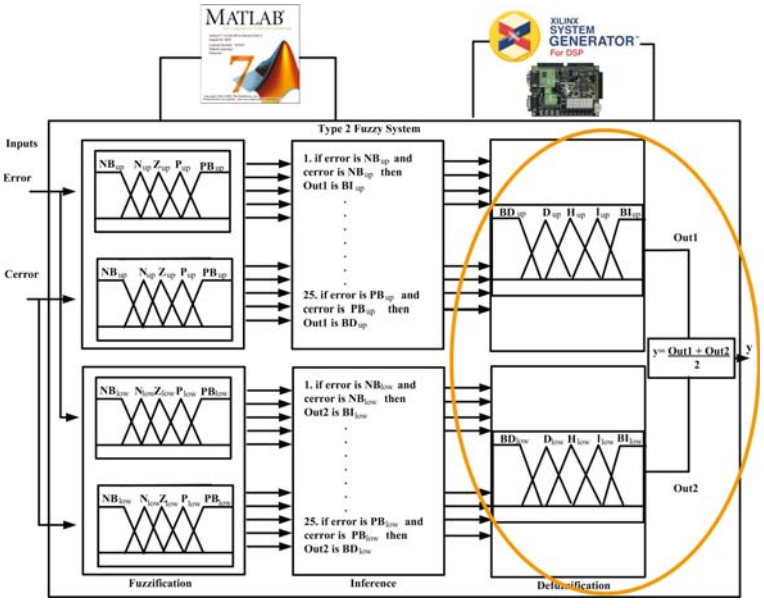

Fig. 1. Proposal for the solution of the defuzzification stage in a type-2 fuzzy system .

With regards to type-2 FIS in FPGA, in [19] is presented the implementation of the Wu-Mendel method in a XC2V3000ff1152-4 FPGA type of the Virtex family, taking advantages of the intrinsic parallelism that these devices offer; also we can find some others works related to this topic in [18, 20].

This work is about the design, test and implementation of the defuzzification stage of a type-2 FIS. The proposal is based on the use of two type-1 FIS, i.e. the average method, to emulate a type-2 FIS, so that the membership functions (MFs) were organized in such a way to emulate the footprint of uncertainty (FOU) of the type-2 MFs, and the final result is obtained as the average of the crisp values obtained in each type-1 FIS; it was used the height method for the type-2 defuzzification stage. In Figure 1 the scheme of the proposed method is shown.

A comparison was made between the control surface of the type-2 average method where the type-1 defuzzification stage is embedded, and the control surface of the type-2 FIS with the Wu-Mendel method.

This paper is organized as follows, Section 2 presents in a general context the type-2 FIS; in Section 3 it is presented the average of two type-1 FIS, as a proposal to avoid the type reduction; Section 4 is dedicated to explain the type-2 defuzzification stage method using VHDL code; Section 5 explains two experiments designed to test the type-2 Defuzzification stage and to verify the accuracy of the results considering that the inputs to this stage were individual values given by hand; Section 6 discusses two experiments and results with a complete type-2 FIS, where the source of the defuzzification stage is VHDL code imported to Simulink through the Xilinx System Generator (XSG) [15] , the fuzzification and the inference stages are models

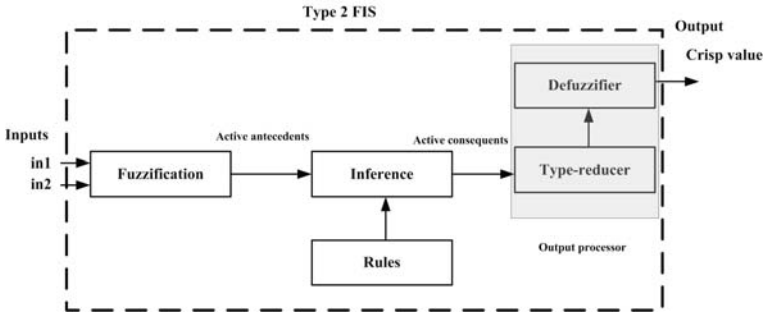

Fig. 2. Type-2 fuzzy system.

from the Simulink. The rules were used for the speed control of a DC motor for the system validation. Finally, Section 6 presents the conclusions of this work.

2 General Contexts of Type-2 FIS

The concept of a type-2 fuzzy set was introduced by Zadeh in 1975 [11], as an extension of the concept of an ordinary fuzzy set in 1965 [12], the membership grade of each of its elements is indeed a fuzzy set in [0, 1], unlike a type-1 set whose membership is a crisp number in [0, 1].

The basics and principles of fuzzy logic do not change from type-1 to type-2 fuzzy sets [4, 5, 6], they are independent of the nature of membership functions, and in general, will not change for any type-n. When a FIS uses at least one type-2 fuzzy set is a type-2 FIS, which is shown in Figure 2 with its components. The structure of the type-2 fuzzy rules, is the same as for the type-1 case because the distinction between them is associated with the nature of the membership functions. Hence, the only difference is that now some or all the sets involved in the rules are of type-2, and as long as any of its antecedents or consequents sets are interval type-2 fuzzy sets, it is called an interval type-2 FIS (IT2FIS) [7, 14] which is the most used. In a type-1 FIS, where the output sets are type-1 fuzzy sets, the defuzzification is performed to get a number, which is in some sense a crisp representation of the combined output sets. In the type-2 case, the output sets are type-2, so it is necessary the extended defuzzification operation to get type-1 fuzzy set at the output. Since this operation converts type-2 output sets to a type-1 fuzzy set, it is called type reduction, and the type-1 fuzzy set obtained is called a type-reduced set, the type-reduced fuzzy set may then be defuzzified to obtain a single crisp number.

2.1 Type-Reduction and Defuzzification in an Interval Type-2 FIS

Five different type-reduction (TR) methods are described in [7, 8]. Center-of-sets, centroid, center-of-sums, and height type-reduction can all be expressed as:

$$Y_{TR}(\mathbf{x'}) = [y_l(\mathbf{x'}), y_r(\mathbf{x'})] \equiv [y_l, y_r]$$

$$[y_l, y_r] = \int_{y^1 \in [y_l^1, y_r^1]} \cdots \int_{y^M \in [y_l^M, y_r^M]} \int_{f^1 \in [\underline{f}^1, \bar{f}^1]} \cdots \int_{f^M \in [\underline{f}^M, \bar{f}^M]} 1 \Big/ \frac{\sum_{i=1}^M f^i y^i}{\sum_{i=1}^M f^i}, \quad (1)$$

where the multiple integral signs denote the union operation [10].

In general, there are no closed-form formula for y_l and y_r; however, Karnik and Mendel [9] have developed two iterative algorithms (known as the Karnik-Mendel or KM Algorithms) for computing these end-points exactly, and they can be run in parallel [10]. Unfortunately this method is computationally costly so it is not well suited for hardware implementation as is the Wu-Mendel method [19].

Wu and Mendel [3] introduced a method to approximate the TR set by minimax uncertainty bounds. These uncertainty bounds are $\underline{y}_l(\mathbf{x}') \le y_l(\mathbf{x}') \le \bar{y}_l(\mathbf{x}')$ and $\underline{y}_r(\mathbf{x}') \le y_r(\mathbf{x}') \le \bar{y}_r(\mathbf{x}')$, where:

$$\bar{y}_l(\mathbf{x}') = min\left\{\frac{\sum_{i=1}^M \underline{f}^i y_l^i}{\sum_{i=1}^M \underline{f}^i}, \frac{\sum_{i=1}^M \bar{f}^i y_l^i}{\sum_{i=1}^M \bar{f}^i}\right\} \quad (2)$$

$$\underline{y}_r(\mathbf{x}') = max\left\{\frac{\sum_{i=1}^M \bar{f}^i y_r^i}{\sum_{i=1}^M \bar{f}^i}, \frac{\sum_{i=1}^M \underline{f}^i y_r^i}{\sum_{i=1}^M \underline{f}^i}\right\} \quad (3)$$

$$\underline{y}_l(\mathbf{x}') = \bar{y}_l(\mathbf{x}') - \left[\frac{\sum_{i=1}^M (\bar{f}^i - \underline{f}^i)}{\sum_{i=1}^M \bar{f}^i \sum_{i=1}^M \underline{f}^i} \times \frac{\sum_{i=1}^M \underline{f}^i (y_l^i - y_l^1) \sum_{i=1}^M \bar{f}^i (y_l^M - y_l^i)}{\sum_{i=1}^M \underline{f}^i (y_l^i - y_l^1) + \sum_{i=1}^M \bar{f}^i (y_l^M - y_l^i)}\right] \quad (4)$$

$$\bar{y}_r(\mathbf{x}') = \underline{y}_r(\mathbf{x}') - \left[\frac{\sum_{i=1}^M (\bar{f}^i - \underline{f}^i)}{\sum_{i=1}^M \bar{f}^i \sum_{i=1}^M \underline{f}^i} \times \frac{\sum_{i=1}^M \underline{f}^i (y_r^i - y_r^1) \sum_{i=1}^M \bar{f}^i (y_r^M - y_r^i)}{\sum_{i=1}^M \underline{f}^i (y_r^i - y_r^1) + \sum_{i=1}^M \bar{f}^i (y_r^M - y_r^i)}\right] \quad (5)$$

Observe that the four bounds in (2)-(5) can be computed without having to perform TR. Wu and Mendel [3] then approximate the TR set, as $[y_l(\mathbf{x}'), y_r(\mathbf{x}')] \approx [(\underline{y}_l(\mathbf{x}') + \bar{y}_l(\mathbf{x}'))/2, (\underline{y}_r(\mathbf{x}') + \bar{y}_r(\mathbf{x}'))/2]$ and compute the output of the IT2 FIS as

$$y(\mathbf{x}') = \frac{1}{2}\left[\frac{\underline{y}_l(\mathbf{x}') + \bar{y}_l(\mathbf{x}')}{2} + \frac{\underline{y}_r(\mathbf{x}') + \bar{y}_r(\mathbf{x}')}{2}\right] \quad (6)$$

So, by using the uncertainty bounds, they obtain both an approximate TR set as well as a defuzzified output [10].

3 Average Type-2 FIS

In cases where the performance of an IT2FIS is important, especially in real time applications, an option to avoid the computational delay of type-reduction, is the Wu-Mendel method [3], which is based on the computation of inner and outer bound

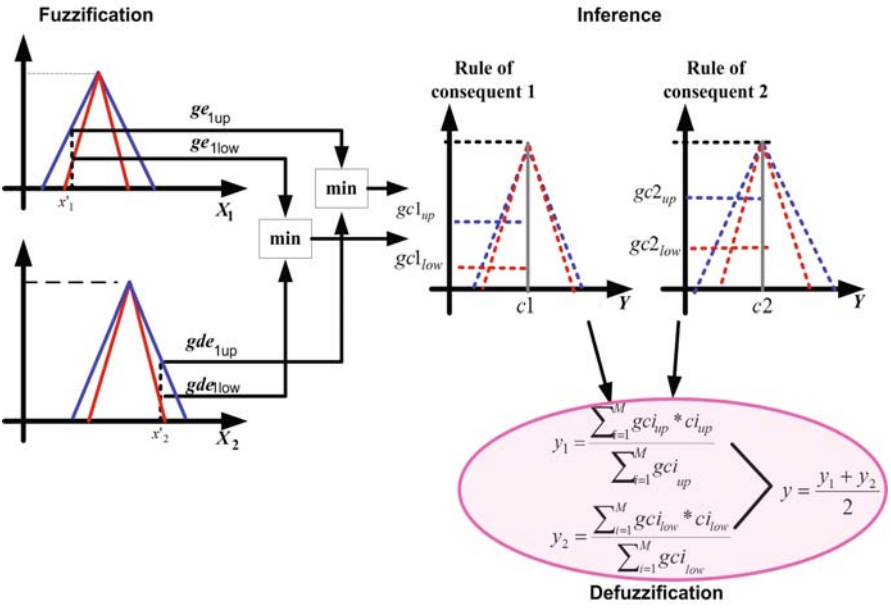

Fig. 3. The fuzzification, the inference and the defuzzification stages in the Average method uses two type-1 FIS.

sets. Another option to improve computing speed in an IT2FIS, is the average of two type-1 FIS method, which was proposed for systems where the type-2 MFs of the inputs and output, have no uncertainty in the mean or center; it is achieved by substituting the IT2FIS with two type-1 FIS, located adequately at the upper and lower footprint of uncertainty (FOU)of the type-2 MFs [25].

For the average method the fuzzification, the inference and the defuzzification stages at each FIS remain identical, the difference is at the output because the crisp value is calculated by taking the arithmetic average of the crisp output of each type-1 FIS, as it is shown in Figure 3, using the height method to calculate the defuzzified crisp output.

4 Type-2 Defuzzification Stage Method Using VHDL Code

In the average method, to achieve the defuzzification, one type-1 FIS is used for the upper bound of uncertainty, and the second FIS for the lower bound of uncertainty. So, as it was explained in Section 3, the defuzzification of a type-1 FIS is used in the average method and it will be explained next.

4.1 Defuzzification in Type-1 FIS

For the explanation of the type-1 defuzzification stage [2], the rule base of a speed control of a DC motor was used, which is presented in Figure 4, along with the

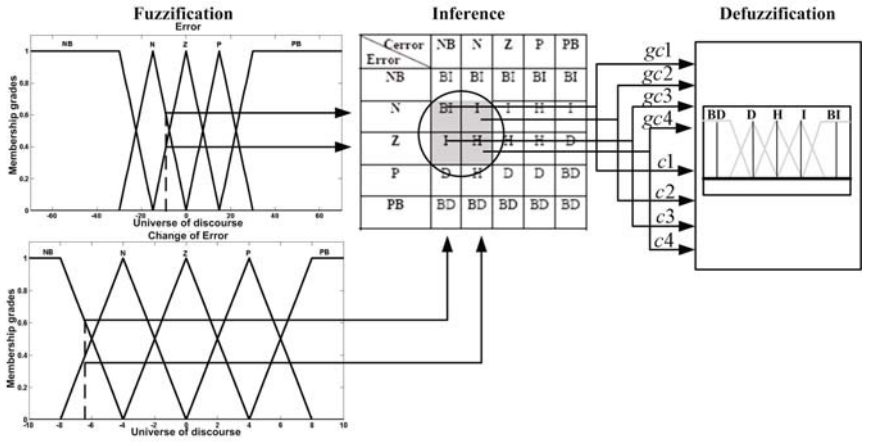

Fig. 4. Rule base for the speed control of a DC motor used for the system validation.

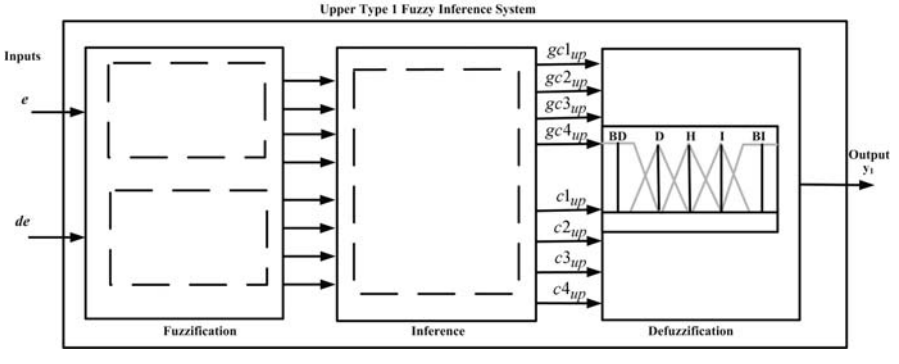

Fig. 5. Defuzzification stage proposal for a FIS.

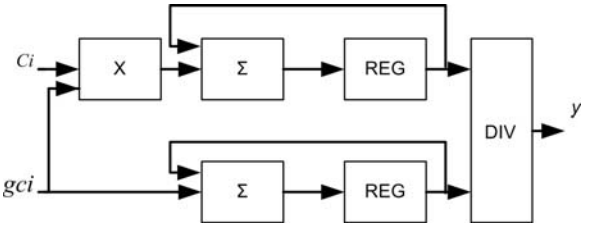

Fig. 6. Block diagram of the height method.

fuzzification and inference stages. In Figure 5 is shown the upper type-1 FIS with two inputs, *e* (error) and *de* (change of error), and one output *y*. Thus the inputs are connected to the fuzzification stage, and considering, for this case, that the method

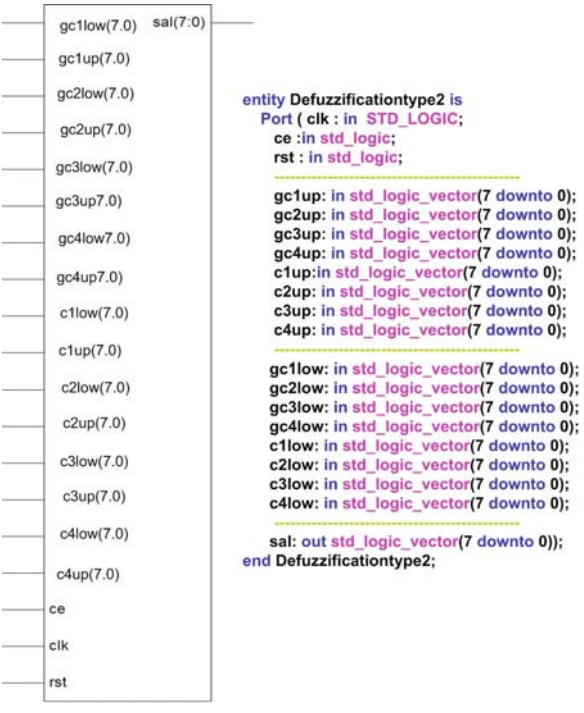

Fig. 7. Entity scheme of the type-2 defuzzification stage.

allows to have a maximum of four active MFs, the fuzzifier delivers two signal for each input, i.e, the membership grade and a tag that identifies the active MF.

The inference engine has eight outputs(considering the four maximum active rules), four are for the firing strengths, the rest correspond to the corresponding linguistic terms of the active consequents. So the output values of the inference engine named as $gc1_{up}$, $gc2_{up}$, $gc3_{up}$, $gc4_{up}$, are the four possible upper firing degrees of the active rules, meanwhile $c1_{up}$, $c2_{up}$, $c3_{up}$ and $c4_{up}$ are the respective center value of the active consequents. The upper part of the defuzzification stage using the gci_{up}, values and ci_{up}'s tags produces a crisp value $y1$ using the height defuzzification method. In the same manner as it is described for the upper defuzzification stage, it is obtained a crisp value $y2$ for the lower part of the defuzzification stage.

The upper defuzzification process using the height method, can be expressed by

$$y1 = \frac{\sum_{i=1}^{M} gci_{up} * ci_{up}}{\sum_{i=1}^{M} gci_{up}}, \tag{7}$$

where:

$y1$ is the crisp output value.
ci_{up} is the center of the active consequent.
gci_{up} is the upper firing degree of the active rule.

The height method was developed in VHDL based on the block diagram shown in Figure 6, where ci is the point of symmetry of the active consequent, gci is the firing degree of the active rule. These two inputs are connected to a block multiplier which corresponds to the part of the numerator in (7); only the addition of the gci, will produce the denominator. Both, numerator and denominator are connected to the block divider. The result of that division is $y1$ [28].

4.2 Implementation of the Type-2 Defuzzification Stage

The design entity of the type-2 defuzzification stage is shown in Figure 7. Such entity was programmed in VHDL to be implemented in a FPGA, but it can be used to simulate the Defuzzification stage without the necessity of designing and implementing any test bench. This entity has 19 inputs and one output. The first eight inputs ($gc1_{low}$, $gc1_{up}$, $gc2_{low}$, $gc2_{up}$, $gc3_{low}$, $gc3_{up}$, $gc4_{low}$, $gc4_{up}$) correspond to the lower and upper firing degrees of the active rules; the next eight inputs ($c1_{low}$, $c1_{up}$, $c2_{low}$, $c2_{up}$, $c3_{low}$, $c3_{up}$, $c4_{low}$ and $c4_{up}$) correspond to the centers of the active consequents. The remaining three input signals are the clock enable, clock, and reset

```
----Initialization----------
        begin
        if (rst='1') then
        conta:=0;
        sal<="00000000";
        den:="0000000010";
        elsif (clk='1' and ce='1')
        then
        den:="0000000010";

---Data Acquisition ---------
        if(conta=0) then
        gra1up:="00"&gc1up;
        et1up:="00"&c1up;
        conta:=conta+1;
        end if;
        if(conta=1) then
        gra1low:="00"&gc1low;
        et1low:="00"&c1low;
        conta:=conta+1;
        end if;
        if(conta=2) then
        gra2up:="00"&gc2up;
        et2up:="00"&c2up;
        conta:=conta+1;
        end if;
        .
        .
        .
```

```
-------Defuzzificaton calculus----------------------

        if (conta=8) then
        divAup:=(gra1up*et1up)+(gra2up*et2up)..
        +(gra3up*et3up)+(gra4up*et4up);
        divBup:=gra1up+gra2up+gra3up+gra4up;
         resul:=divide(divAup,divBup);
        resulup:="000000000000"&resul(1)(7 downto 0);
        conta:=conta+1;
        end if;
        if(conta=9)then
        divAlow:=(gra1low*et1low)+(gra2low*et2low)..
        +(gra3low*et3low)+(gra4low*et4low);
        divBlow:=gra1low+gra2low+gra3low+gra4low;
        resul:=divide(divAlow,divBlow);
        resullow:="000000000000"&resul(1)(7 downto 0);
        conta:=conta+1;
        end if;

---------Average-----------------------------------------------

        if (conta=10) then
        resultemp:=resulup+resullow;
        resul:=divide(resultemp,den);
        sal<=resul(1)(7 downto 0);
        conta:=0;
        end if;
```

Fig. 8. VHDL code for the type-2 Defuzzification stage.

(ce, clk, rst) that allow the simulation of VHDL code in Simulink environment. The output port "sal", delivers the crisp value of the type-2 defuzzification stage.

In Figure 8 the VHDL code of the defuzzification stage for a type-2 FIS is presented, it is divided in four sections: Initialization, data acquisition, defuzzification computation, and the average calculus. In the first section the output port is initialized, the internal count register, which allows to carry out the data acquisition, besides determines the vector of the denominator of the division that will realize the average of the obtained results of each type-1 defuzzification stage. In the second section, the input data is obtained, i.e. the four possible firing strengths with the central values of the active consequents for each type-1 defuzzification stage. The input values are stored in registers, starting with the couple ($gc1_{up}$, $c1_{up}$ and $gc1_{low}$, $c1_{low}$) until the couple ($gc4_{up}$, $c4_{up}$ and $gc4_{low}$, $c4_{low}$) which correspond to the upper and lower firing strength and their respective activated consequent. The defuzzification calculus component, takes each input data couple and using aritmethic operations like addition, multiplication and division, computes the average of the two type-1 defuzzification stages with the height method, and sent it to the output port "sal". After that, the count register is initialized again, ready to the acquisition of new input data, and carry on the new defuzzification calculus in a cyclical process.

5 Test of the Type-2 Defuzzification Stage

To test the type-2 defuzzification stage, two experiments were conducted. The comparison was made between the defuzzification stage developed in VHDL code for several bits of resolution, and the Wu-Mendel method [3] programmed in Matlab code and whose block diagram is shown in Figure 9.

In order to achieve the experiments to test the defuzzification stage three main different software tools were used: The Simulink from Mathwork which is a practical high-level design and simulation tool because it provides a flexible design and simulation platform to test and correct designs at high level; the Xilinx Integrated Software Environment (Xilinx ISE) which is a Hardware Description Language (HDL) design software suite that allows taking designs through several steps in the ISE design flow finishing with final verified modules that can be implemented in a hardware target such a Field Programmable Gate Array (FPGA); and the Xilinx System Generator (XSG), which is a DSP design tool that enables the use of Simulink for FPGA design. This tool is very important because it allows generating VHDL code from the System Generator Simulink modules; and visceversa, VHDL modules can be included in the Simulink design platform by importing the VHDL code in a System Generator "Black box", the VHDL code in converted to Matlab functions.

As a first step the development of the stage was carried out using VHDL programming in the ISE Xilinx v8.2i [15], simulated and tested in Simulink/Matlab using the XSG. The experiments were based on the proposed type-2 MFs for the output variable, shown in Figure 10. This variable has five MFs, two trapezoidal and three triangular, on the universe of discourse proposed. As the height method was used as a defuzzification method, in fact there were used singletons to represent the central value of each MF.

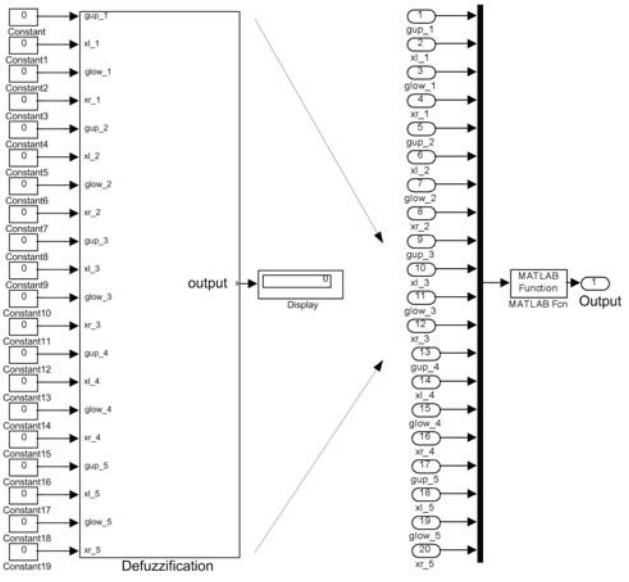

Fig. 9. Output using the Wu-Mendel method.

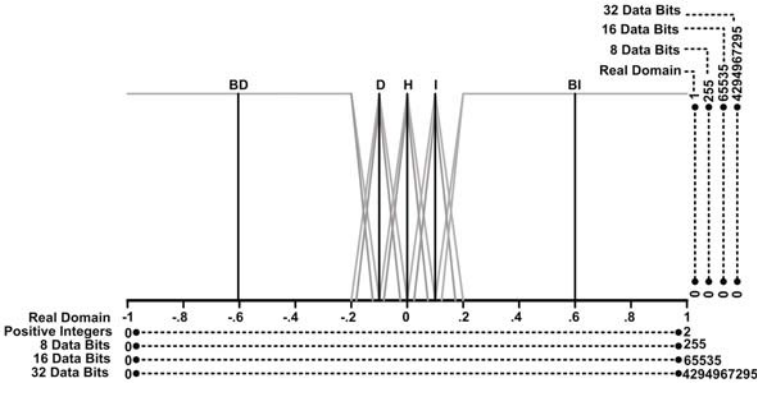

Fig. 10. Output variable Membership Functions.

The experimental defuzzification stage coded in VHDL using the average of two type-1 fuzzy systems is presented in Figure 11. The stage was coded in VHDL and simulated in Simulink/Matlab through the XSG tool. The stage was tested for 8, 16, and 32 bits resolution.

Experiment 1. In this experiment it is assumed that only two MFs are activated, D and H of Figure 10, by the inference stage . The values for their firing strenghs are assumed to be D_{up}=0.7 and D_{low}=0.4; H_{up}=0.6 and H_{low}=0.2. The output was

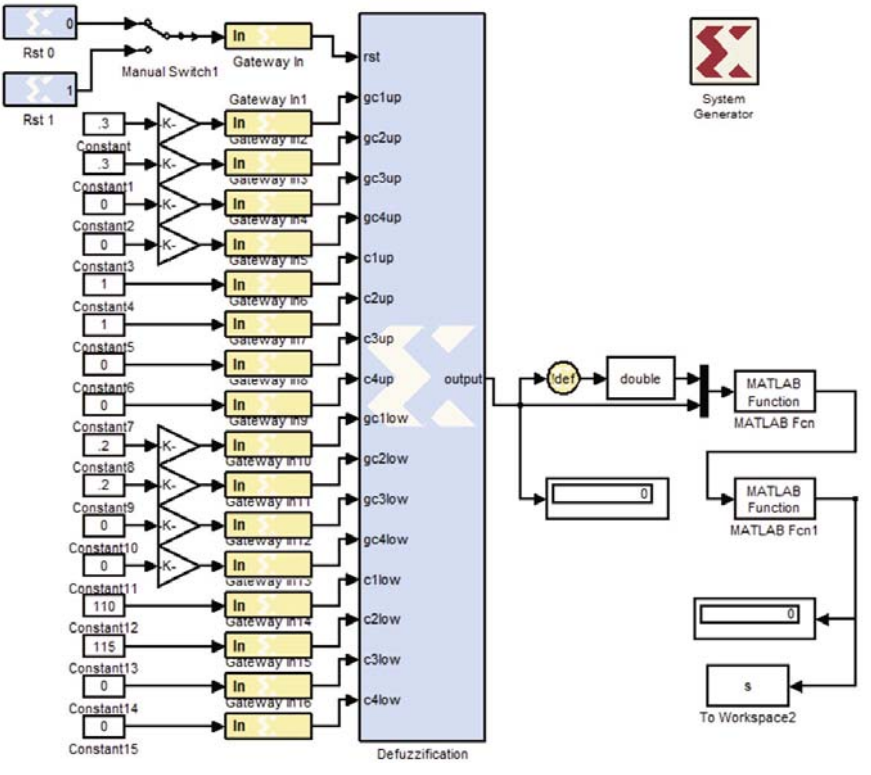

Fig. 11. Experimental design for the type-2 Defuzzification stage.

Table 1. Results of experiment 1 for the type-2 defuzzification stage.

Type-2 MF	Resolution of Defuzzification	Bits value	Output numeric value
D_{up}=0.7 D_{low}=0.4	8 bits	119	-0.06667
	16 bits	30793	-0.06026
H_{up}=0.6 H_{low}=0.2	32 bits	2018083991	-0.06028
	Wu-Mendel	—	-0.05889

obtained for 8, 16 and 32 bits of resolution using for each case the average method and the height as a defuzzifier. For example in the case of 8 bits of resolution, it was obtained a 119 value equivalent to -0.06667 in the real domain. The result obtained using the Wu-Mendel method coded in Matlab, was -0.05889. These results are shown in Table 1.

Experiment 2. In this case three MFs are activated, D, H and I of Figure 10, by the inference stage. The values for their firing strenghs are assumed to be D_{up}=0.5 and D_{low}=0.4; H_{up}=0.3 and H_{low}=0.2; finally I_{up}=0.2 and I_{low}=0.1. After the conversion

Table 2. Results of experiment 2 for the type-2 defuzzification stage.

Type-2 MF	Resolution of Defuzzification	Bits value	Output numeric value
D_{up}=0.5 D_{low}=0.4	8 bits	122	-0.04314
	16 bits	31573	-0.03645
H_{up}=0.3 H_{low}=0.2	32 bits	2069253886	-0.03643
I_{up}=0.2 I_{low}=0.1	Wu-Mendel	—	-0.0366

for the different resolutions The output was obtained for 8, 16 and 32 bits of resolution using for each case the average method and the height as a defuzzifier. In the case of 16 bits of resolution it was obtained a 31573 value equivalent to -0.03645 in the real domain. The result obtained using the Wu-Mendel method coded in Matlab, was -0.0366, almost the same as the obtained with a 16 bits of resolution. These results are shown in Table 2.

As can be seen in Tables 1 and 2, the results obtained with the Wu-Wendel method developed with Matlab code are similar to the ones obtained with the proposed method and 16 bits resolution of the data inputs.

6 Testing the Type-2 Defuzzification Stage in the Fuzzy System

Once the simulation of the VHDL type-2 defuzzification stage has been done, the next step is to test it in a type-2 FIS with the average of two fuzzification and inference stages, programmed using the appropriated Matlab function from the Fuzzy Matlab Toolbox. In the same way as in the type-1 defuzzification stage, having two parallel type-1 fuzzy systems the output is obtained as the average of the crisp values of each system. There were realized three experiments, the tests were done to verify the operation of the whole type-2 system in Simulink/Matlab using the Xilinx System Generator library. Comparisons were made between the results obtained with 8, 16 and 32 bits for the data inputs for the defuzzification stage, and those obtained with the Wu-Mendel method in Matlab code, recalling that in this one was used floating point in the input and output data. For the validation of the defuzzification stage in the compound type-2 FIS, the experiments undergo the same input conditions, with the purpose of making comparisons between the obtained results. The MFs proposed for the two input variables and for the output, are shown in Figure 12 and in Figure 10, respectively.

In Figure 13 is presented the whole Simulink model of the type-2 FIS, in which the two first stages, fuzzification and inference, were designed in Simulink/Matlab blocks, while the third one, defuzzification, using VHDL codification and imported to Simulink through the Xilinx System Generator.

Experiment 3. The type-2 FIS developed has two inputs, error and change of error, and one output. Assuming the following values for the inputs: error=-20, change of error=3, after the simulation of both type-2 FIS, the results for the 8, 16, and 32 bits

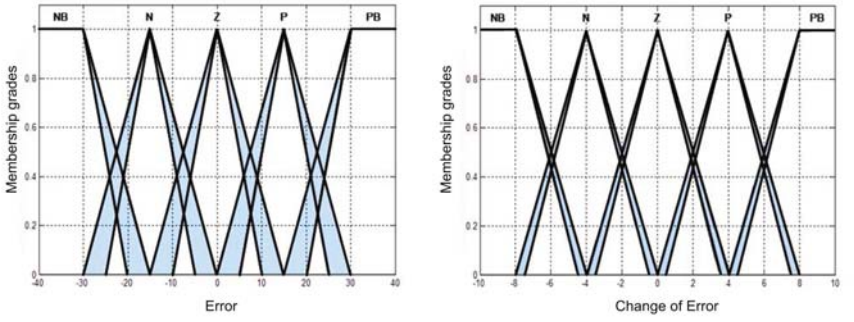

Fig. 12. Type-2 MF for the inputs of the Type-2 FIS.

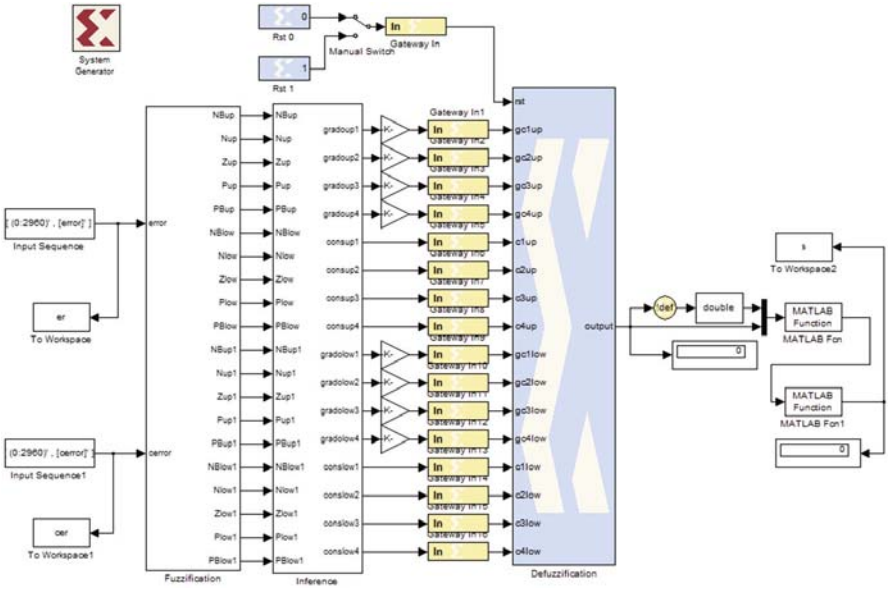

Fig. 13. Test of the type-2 defuzzification stage in a Type-2 FIS.

input data for the test of the type-2 defuzzification stage in the type-2 FIS, as well for the Wu-Mendel type-2 FIS in Matlab Model are shown in Table 3.

Experiment 4. When the values of the two inputs of the type-2 FIS were assumed to have the following values: error=10, change of error=-7, after the simulation of both type-2 FIS, the results for the 8, 16, and 32 bits input data for the test of the type-2 defuzzification stage in the type-2 FIS, as well for the Wu-Mendel type-2 FIS in Matlab Model are shown in Table 4.

The results obtained for the type-2 FIS using 16 bits for the input data are very similar to the results obtained with the type-2 FIS with the Wu-Mendel method.

Table 3. Results of experiment 3 for the type-2 defuzzification stage in the type-2 FIS, for error=-20 and cerror=3

Data Type	Bits value	Output numeric value
8 bits	139	-0.0902
16 bits	35716	-0.08998
32 bits	23640757176	-0.09
Wu-Mendel in Matlab	—	-0.09732

Table 4. Results of experiment 4 for the type-2 defuzzification stage in the type-2 FIS, for error=-20 and cerror=3

Data Type	Bits value	Output numeric value
8 bits	119	-0.06667
16 bits	30692	-0.06334
32 bits	2011476350	-0.06333
Wu-Mendel in Matlab	—	-0.06945

Experiment 5. To make a comparison between the behavior of the type-2 FIS with the defuzzification stage coded in VHDL, and the type-2 FIS using the Wu-Mendel method, a control surface was obtained for both of the fuzzy systems. To perform this experiment, it was necessary to design a Matlab function capable of generating two vectors containing all possible combinations for the inputs of each type-2 FIS, and then get the answer and generate the control surface. In Figure 14 is presented the control surface for the type-2 FIS with the defuzzification coded in VHDL, and in Figure 15 the control surface for the type-2 FIS with the Wu-Mendel method. It

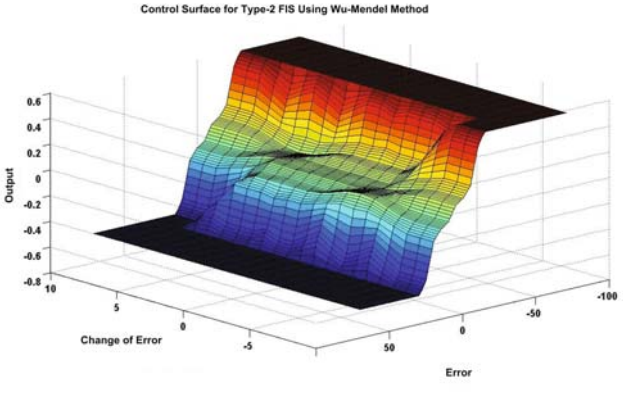

Fig. 14. Control surface using the Wu-Mendel method in the type-2 FIS.

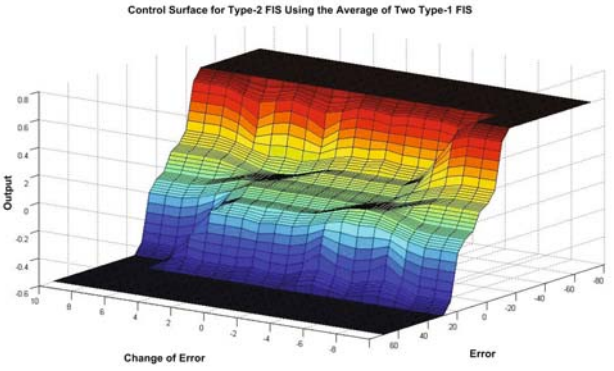

Fig. 15. Control surface using the Average of two type-1 FIS method

can be seen that the difference in both surfaces is minimum, although the input data for the latter are of floating point type, which allows to obtain more exact values.

7 Conclusions

The software implementations of fuzzy systems have been the most widely adopted, the highly flexibility is one of their characteristics, however the response time is limited by the inherent sequential execution of the programs, which is not convenient in real time applications, where the inference speed is an important factor. For this reason the number of applications using fuzzy systems into an FPGA is increasing, because one of its main advantages is the parallel processing, the low cost , low power and area consumption, etc. The methodology proposed in this paper in the design of the type-2 inference engine using a hardware description language, give a more practical and flexible model since the process of updating any change in the rule base can be made easily. So this is an alternative to quickly achieve both goals, since the developed code to describe the hardware, i.e. the system model, can be used with any modification to simulate the system and make the system implementation in the final target, for example in an FPGA.

Acknowledgements. The authors thank Comisión de Operación y Fomento de Actividades Académicas del I.P.N., Instituto Tecnológico de Tijuana for supporting our research activities, and CONACYT.

References

1. Lago, E., Hinojosa, M.A., Jimńez, C.J., Barriga, A., Sánchez-Solano, S.: FPGA Implementation of Fuzzy Controllers. In: Proc. XII Conf. on Design of Circuits and Integrated Systems (DCIS 1997), Sevilla, November 1997, pp. 715–720 (1997)

2. Lizárraga, G., Sepúlveda, R., Montiel, O., Castillo, O.: Modeling and Simulation of the Defuzzification stage using Xilinx System Generator and Simulink. Soft Computing for Hybrid Intelligent Systems, vol. 154, pp. 333–343. Springer, Heidelberg (2008)
3. Wu, H., Mendel, J.M.: Uncertanty Bounds and Their use in the Design of Interval type-2 Fuzzy Logic Systems. IEEE Transactions on Fuzzy Systems 10, 1–16 (2002)
4. Mendel, J.M.: Type-2 Fuzzy Sets and Systems: an Overview. IEEE Computational Intelligence Magazine 2, 20–29 (2007)
5. Mendel, J.M.: Type-2 Fuzzy Sets: Some Questions and Answers. IEEE Computer Society Press, Los Alamitos (2003)
6. Mendel, J.M., Bob John, R.I.: Type-2 Fuzzy Sets Made Simple. IEEE Transactions on Fuzzy Systems 10, 117–127 (2002)
7. Mendel, J.M.: Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Prentice-Hall, Upper-Saddle River (2001)
8. Karnik, N.N., Mendel, J.M., Liang, Q.: Type-2 Fuzzy Logic Systems. IEEE Transactions on Fuzzy Systems 7, 643–658 (1999)
9. Karnik, N., Mendel, J.M.: Centroid of a Type-2 Fuzzy Set. Information Sciences 132, 195–220 (2001)
10. Mendel, J.M., Hagras, H., John, R.I.: Standard Background Material About Interval Type-2 Fuzzy Logic Systems That Can Be Used By All Authors
11. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning. Information Sciences 8, 199–249 (1975)
12. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
13. L.E., Jiménez, C.J., López, D.R., Sánchez-Solano, S., Barriga, A.: XFVHDL: A Tool for the Synthesis of Fuzzy Logic Controllers. Design Automation and Test in Europe, 102–107 (1998)
14. Liang, Q., Mendel, J.M.: Interval Type-2 Fuzzy Logic Systems: Theory and Design. IEEE Trans. on Fuzzy Systems 8, 535–550 (2000)
15. Manual of Xilinx System Generator, `http://www.xilinx.com`
16. Patyra, M.J., Janos, L., Koster, K.: Digital Fuzzy Logic Controller:Design and Implementation. IEEE Transactions on Fuzzy Systems, vol. 4(4) (November 1996)
17. Melgarejo, M.: Desarrollo de un Sistema de Inferencia Difusa sobre FPGA, Universidad Distrital Francisco José de Caldas, Centro de Investigación y Desarrollo Cientifico, Esfera Editores, Bogotá, Colombia (2003)
18. Melgarejo, M., Peña-Reyes, C.A.: Implementing Interval type-2 Fuzzy Processors. Computational Intelligence Magazine, IEEE 2, 63–71 (2007)
19. Melgarejo, M.A., Carlos, R., Peña-Reyes, A.: Hardware architecture and FPGA implementation of a type-2 fuzzy system. In: Proceedings of the 14th ACM Great Lakes symposium on VLSI, pp. 458–461 (2004)
20. Melgarejo, M.A., Garcia, R.A., Peña-Reyes, C.A.: Pro-Two: a hardware based platform for real time type-2 fuzzy inference. In: Proceedings of 2004 IEEE International Conference on Publication Fuzzy Systems, vol. 2, pp. 977–982 (2004)
21. Cirstea, M.N., Dinu, A., Khor, J.G., McCormick, M.: Neural and Fuzzy Logic Control of Drives and Power System, Newnes (2002)
22. Montiel, O., Maldonado, Y., Sepúlveda, R., Castillo, O.: Simple tuned fuzzy controller embedded into an FPGA. In: 2008 NAFIPS Conference Proceedings, pp. 1–6 (2008)
23. Montiel, O., Olivas, J., Sepúlveda, R., Castillo, O.: Development of an Embedded Simple Tuned Fuzzy Controller. In: Zurada, J.M., Yen, G.G., Wang, J. (eds.) WCCI 2008. LNCS, vol. 5050, pp. 555–561. Springer, Heidelberg (2008)
24. Vuong, P.T., Madni, A.M., Voung, J.B.: VHDL Implementation for a Fuzzy Logic Controller. In: World Automation Congress, WAC apos; 2006, pp. 1–8 (2006)

25. Sepúlveda, R., Castillo, O., Melin, P., Díaz, A.R., Montiel, O.: Experimental Study of Intelligent Controllers under Uncertainty using type-1 and type-2 Fuzzy Logic. Information Sciences 177, 2023–2048 (2007)
26. Poorani, S., Urmila Priya, T.V.S.: FPGA Based Fuzzy Logic Controller for Electric Vehicle. Journal of the Institution of Engineers, Singapore 45(5) (2005)
27. Iregui, S., Linares, D., Melgarejo, M.: Performance Evaluation of Fuzzy Operators for FPGA Technology. In: NAFIPS 2008, New York (2008)
28. Sánchez-Solano, S., Cabrera, A., Jiménez, C.J., Brox, P., Baturone, I., Barriga, A.: Implementación sobre FPGA de Sistemas Difusos Programables. In: IBERCHIP, Workshop IBERCHIP La Habana, Cuba (2003)
29. Sánchez-Solano, S., Senhadji, R., Cabrera, A., Baturone, I., Jiménez, C.J., Barriga, A.: Prototyping of Fuzzy Logic Based Controllers Using Standard FPGA Development Boards. In: 13th IEEE International Workshop on Rapid System Prototyping on Volume, pp. 25–32 (2002)
30. Sánchez Solano, S., Barriga, A., Jiménez, C.J., Huertas, J.L.: Design and Application of Digital Fuzzy Controllers. In: Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEED 1997), Barcelona, Spain, vol. 2, pp. 869–874 (1997)