Digital Fuzzy Inference Engine Simulator

Antonio Hernández Zavala[1], Oscar Camacho Nieto[1], Ildar Batyrshin[2], Osvaldo Espinosa Sosa[1]

[1]Instituto Politécnico Nacional, [2]Instituto Mexicano del Petróleo
antonioh_z@hotmail.com, oscarc@cic.ipn.mx, batyr@imp.mx, espinosa@cic.ipn.mx

Abstract. Fuzzy control has growing to many applications in the course of time since it was developed at early 60's. Recent works as they increase in complexity require every time more calculations; this is why usage of specific fuzzy hardware is becoming more suitable to satisfy processing time demands. It has been noticed that conventional fuzzy processors and simulators use operations such as multiplications and divisions over an input space given by the number of resolution bits used, this is reflected in the total time consumed to give a crisp result out from an input. To reduce time consumed we avoid usage of these equations by replacing them with adders and shifting. As representation of real functions in binary terms is really not continuous, membership functions are not represented at all elements, they are discretized into m quantization levels called $\alpha - levels$ which are on dependence of the number of resolution bits used for membership universe space and is independent of bits used for input space. There is no simulator that let us visualize how inference procedure is performed in terms of integer numbers as they are used by the computer architecture, on this work a "Digital Fuzzy Inference Engine Simulator (DiFES)", which is realized to satisfy fuzzy hardware design demands according to mentioned representation, is presented in order to visualize inference procedure at a hardware equations level to make easier this design process.

Keywords: Fuzzy Logic, Fuzzy Sets, Defuzzification, Discrete Numbers, Quantization, Inference Simulation.

1 Introduction

Since the first application of fuzzy logic to control a steam engine realized by Mamdani at 70's [1], several applications have grown up, among this we can mention Sugeno's work on an automatic parking car [2], high performance applications on water treatment [3][4], nuclear reactors [5][6], robotics[7], pattern recognition [8] and physics experiments [9], among others.

Fuzzy logic has been widely accepted because it can be used to efficiently translate human knowledge into control rules for different applications. It has been shown that a fuzzy inference controller is robust and gave better results than conventional controllers. There are researchers that have shown all advantages that fuzzy logic controllers offer, considering them as universal aproximators [10][11][12][13].

Computer Engineering.
Research in Computing Science 30, 2007, pp. 107 - 118

While applications are extended to more complex processes that require faster processing speed, they have turned to use specific inference hardware. First digital fuzzy processor was realized by Watanabe and Togai at 1985. Since then, many interesting architectures have been presented [14], [15], [16], [17] y [18]. Each of them has improved the processor features, obtaining every time less complicated hardware that realizes inferences faster, consuming less processing time and decreasing number of hardware resources.

With FPGA boom, because of its binary reprogrammable capabilities, some researchers have used this dispositive to implement fuzzy hardware [19][20][21][22][23], on this works digital circuitry is adapted according to the mathematics of fuzzy logic. The tendency is to avoid as possible time consuming operations as multiplications and divisions, these two operands need many instruction cycles to give a result.

Actually there is no simulator system where all steps required to implement digital algorithms of fuzzy logic are visualized, on this paper we give an approach to this, first, we give an introduction to fuzzy control where general controller is shown to explain its blocks digital realization on consequent chapters. A simulator on C++Builder programming language, was realized with equations shown in content, to finally give some results and conclusions.

2 Fuzzy Control

The parts which compose a fuzzy controller are shown on Fig. 1; outside the dotted line is the crisp type environment of the controller. Fuzzifier and defuzzifier are translators between crisp and fuzzy numbers at input and output respectively; rule base contains all the rules that describe system behaviour from where inference decides which rules use to make a decision. Each block is commented on following parts.

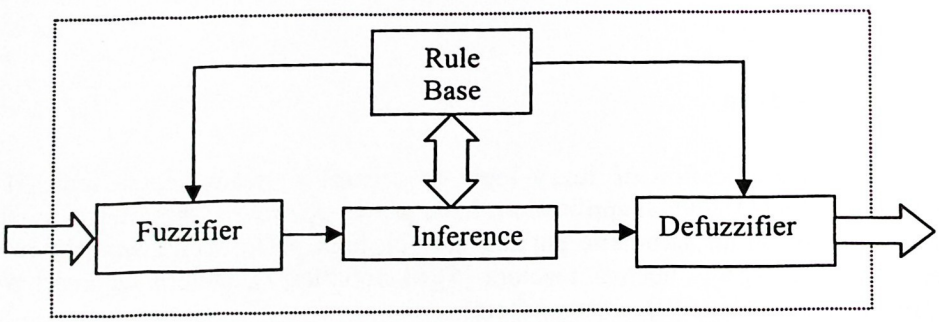

Fig. 1. Fuzzy controller blocks.

2.1 Fuzzifier

First block from left on Fig. 1 corresponds to the fuzzifier stage where the real input value is mapped into the corresponding universe of discourse, on binary numbers this is called discretization or quantization of input universe into m number of $\alpha - levels$ [24] and a membership value is assigned to each of n points in x.

As mentioned before, the operator fuzzifier can take different forms, on this document $\alpha - levels$ are used to assign different membership values to a membership function in the following way:

Let M be a trapezoidal membership function, with $x_i \in X; i = 0,1,\ldots,n, X \in [0, 2^n - 1]$; the number of points in x axis, and m discretization levels that give the resolution on y axis. Let's assume that we have the base of M with height equal to 0 or α_0, in the range from $x=0$ and $x=n$; calling these two points initial ($x_0^{\alpha_0}$) and final ($x_f^{\alpha_0}$) values respectively, this segment will be called the base and its length is given by the following equation (1):

$$Lenght_{\alpha_0} = x_f^{\alpha_0} - x_0^{\alpha_0} \tag{1}$$

As we are using integer numbers, α_0 will have height equal to 0, then its area is equal to 0, this is an interval defined by the length of the segment and is used just to know if input x corresponds to this function as on (2).

$$IF\ x \in \left\lfloor x_0^{\alpha_0}, x_f^{\alpha_0} \right\rfloor THEN\ x \in M \tag{2}$$

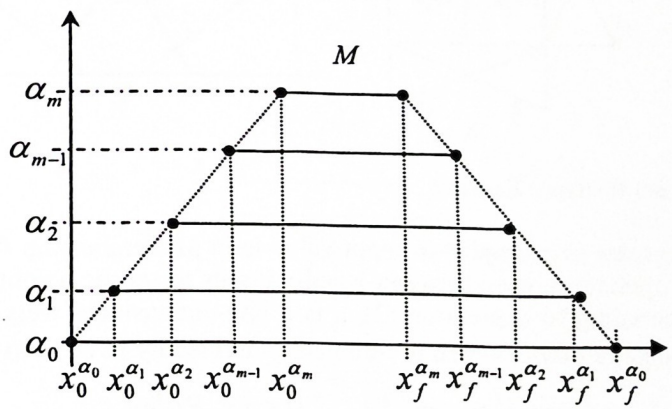

Fig. 2. Discretization of a trapezoidal membership function into $\alpha - levels$.

Every $\alpha - level$ for a membership function will be then defined by the following four parameters: M is the name of the function and is used to group $\alpha - levels$, $x_0^{\alpha_m}$ is the initial point and $x_f^{\alpha_m}$ is the final point of the interval; α_m is the height or

membership value for all the elements in $\left[x_0^{\alpha_m}, x_f^{\alpha_m}\right]$. Fig. 2 shows a discretization of a membership function with m $\alpha-levels$ and $n = x_f^{\alpha_0}$ points on the space of X. The whole M membership function is then defined by the set of all its $\alpha-levels$ which describe it totally.

To evaluate a point on M we check first (2) and if true we can test on which $\alpha-levels$ of M this point matches and keep record of which was the higher membership value, and the name of the function. With this information for each function activated we can perform inference procedure, which will be shown on the following section.

2.2 Mamdani Inference

Inference procedure used here is Mamdani form corresponds to middle blocks on Fig.1, exemplified on Fig. 3 with a two input-one output fuzzy system, with two and three membership functions respectively; $x \in X, y \in Y, z \in Z$ to form truth table, there is an output function activated for each pair of inputs. When two points (x, y) are evaluated as input, they can be operated with any t-norm or t-conorm to realize conjunction or disjunction, this gives as result the height on α and the membership function to be evaluated.

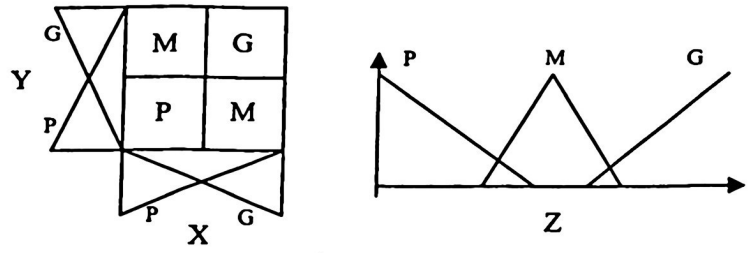

Fig. 3. Mamdani Inference Example

Let us consider as example: an input value is in M membership function when (2) is true, min operator is used between x and y inputs to set the height of the rule to be evaluated according to expression (3), z is a one-dimensional weight vector used to obtain the shape of output which is completely defined by its $\alpha-levels$.

$$z_i = \min\left(\alpha_{x_i}, \alpha_{y_i}\right); \qquad 0 \le i \le Rules_{TOT} \tag{3}$$

Next step is to create homogeneous intervals, this is realized using *min* and *max* operators on each level for each activated function obtaining the length for output function, this is repeated for each $\alpha_i < \alpha_{\max}$ until reaches maximum membership value $\alpha_{\max}$, the initial and final points are obtained with (4). The resultant set of $\alpha-levels$ is aggregated to obtain the area, this is realized with (5). Up to now we

have described the shape of resultant function, which is next, is to defuzzify and obtain a crisp output in the following section.

$$\alpha_i^z \begin{cases} z_0^{\alpha_i^z} = \min\left(z_0^{\alpha_i^{PP}}, z_0^{\alpha_i^{PG}}, z_0^{\alpha_i^{GP}}, z_0^{\alpha_i^{GG}}\right) \\ z_f^{\alpha_i^z} = \max\left(z_0^{\alpha_i^{PP}}, z_0^{\alpha_i^{PG}}, z_0^{\alpha_i^{GP}}, z_0^{\alpha_i^{GG}}\right) \end{cases} \tag{4}$$

$$A_Z = \sum_{i=0}^{\alpha_{max}} \left(z_f^{\alpha_i^z} - z_0^{\alpha_i^z}\right)\alpha \tag{5}$$

2.3 Defuzzification

There exists many methods to defuzzify, right block on Fig. 1, all of them satisfy different precision needs and speed requirements, most commonly used on literature for digital implementations are center of gravity COG [25], center of average [26], among others. A good research on this is found at [27], where new methods are introduced and compared with previous ones. Another form of fast defuzzifier is on [28].

We have distinguished that all of this methods need at least $k-1$ iterations according to the input spaces given by 2^n where n is the number of bits used. We will use Center of Slice Area Average defuzzifier, on which we need α_{max} iterations and sum midpoints of every α_i, then divide them between the maximum α reached, this is expressed by (6). As example on Fig. 4, there is a case where $m=4$ here, the 4 centers are shown for every α_i, and output is marked as COSAA.

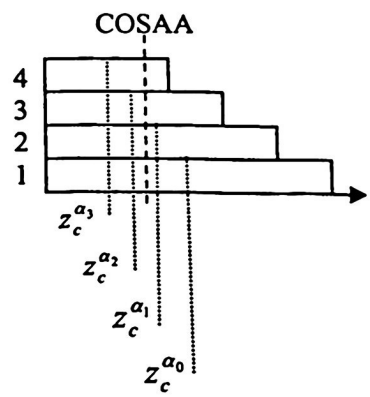

Fig. 4. Center of Slice Area Average

$$COSAA = \frac{\sum_{i=0}^{\alpha_{max}} \left(\frac{\left(x_f^{\alpha_i} - x_0^{\alpha_i}\right)}{2} + x_0^{\alpha_i} \right)}{\alpha_{max}} \tag{6}$$

3. Fuzzy Inference Simulator (DiFES)

Methodology presented here was simulated on a C++Builder program on which every inference stage was implemented according to equations (1-6). Here two inputs with two membership functions each and one output with three membership functions as on Fig. 3 is discretized at 2 bits for membership i.e. four $\alpha - levels$, and 4 bits for x axis, i.e. 16 values on the input universe.

DiFES Simulator is divided on four sections: Input Functions, Output Functions, Inference Evaluation and Defuzzification is the last section, following sections detail them.

3.1 Input Membership Functions

First section corresponds to the input functions for both inputs x, y shown on Fig. 5 where correspondent functions are for x input: SMALLx and BIGx, for y input are SMALLy and BIGy, they are shown separately with the objective of visualize them in a better way. Numbers that compose each function are used to identify its membership value, which is at least 1 and at most 4. These functions are identical for each input and are defined by equation (7) for SMALL and (8) for BIG, where can be noticed that equations correspond to trapezoids defined by intervals and are overlapped each other. As can be seen and as we are using two bits to represent membership value a continuous function can not be represented because of discretization.

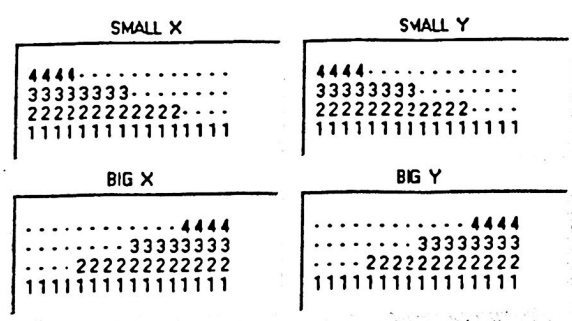

Fig. 5. Input Membership Functions for x and y.

$$\mu_{SMALL} = \begin{cases} \alpha_{MAX}, & [0,3] \\ -\frac{\alpha_{MAX}}{12}x + 4, & [3,15] \end{cases} \quad (7)$$

$$\mu_{BIG} = \begin{cases} \frac{\alpha_{MAX}}{12}x, & [0,12] \\ \alpha_{MAX}, & [12,15] \end{cases} \quad (8)$$

Following code is realized to create the set of $\alpha - levels$ that define input membership functions (7) (8) for the variable x.

```
For (int a=0; a<=3;a++) //**********SMALLx construction
        {
         MF[a].function=1;
         MF[a].start=0;
         MF[a].end= 15 - 4*a;
         MF[a].area = (MF[a].end - MF[a].start) +1;
         MF[a].alfalevel=a+1;
        }
For (int a=4; a<8;a++) //*********BIGx Construction
        {
         MF[a].function=2;
         MF[a].start=4*(a-4);
         MF[a].end=15;
         MF[a].area = ((MF[a].end - MF[a].start)+1);
         MF[a].alfalevel=a-3;
        }
```

3.2 Output Membership Function

Second part corresponds to output membership functions for output variable z, they are SMALLz, MEDIUMz and BIGz; these functions are defined by the equations (9-11), Fig. 6 shows simulator output membership functions, according to the equations mentioned. Discretization is like on previous section at 2 bits for membership and 4 bits for input space.

$$\mu_{SMALLz} = \begin{cases} \alpha_{MAX}, & [0,2] \\ -\frac{\alpha_{MAX}}{12}x + 4, & [2,8] \end{cases} \quad (9)$$

$$\mu_{MEDIUM_Z} = \begin{cases} \frac{\alpha_{MAX}}{3}x - 4, & [4,7] \\ \alpha_{MAX}, & [7,8] \\ -\frac{\alpha_{MAX}}{3}x + 11, & [4,7] \end{cases} \tag{10}$$

$$\mu_{BIG_Z} = \begin{cases} \frac{\alpha_{MAX}}{6}x - 4, & [7,13] \\ \alpha_{MAX}, & [13,15] \end{cases} \tag{11}$$

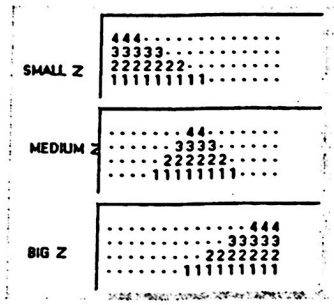

Fig. 6. Output Membership Functions

3.3 Inference Evaluation

Third section of simulator is where crisp inputs are introduced for x and y over the input universe (4 bit), on top of Fig. 7, these values are evaluated on input membership functions for each when pushing EVALUATE button.

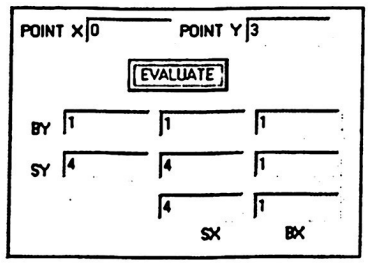

Fig. 7. Inference Evaluation

Down evaluation button, there are two columns labelled SX and BX, they correspond to membership functions SMALLx and BIGx respectively; the rows are the same for y input, SY, BY correspond to SMALLy and BIGy respectively. Each of

mentioned labels has a text box where membership value activated for that respective input is shown, these values are used to generate rule weight matrix with the intersection points using *min* operator as on (3).

3.4 Defuzzification

After stages mentioned, last to do is to obtain a resultant membership function, by means of aggregation operator (5), and to get a crisp output from it using a defuzzifier (6). Where figure is the resultant aggregated function, and COSAA is the crisp type output value obtained from center of slice area average method. On top of Fig. 8, there is aggregated output membership function which is used to obtain defuzzified output COSAA as a crisp value over the output universe.

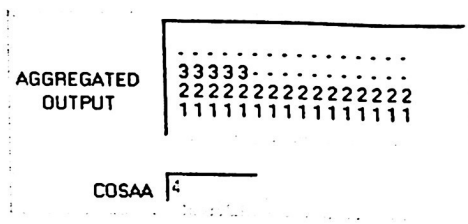

Fig. 8. Aggregation and defuzzification of output membership function

4 Results

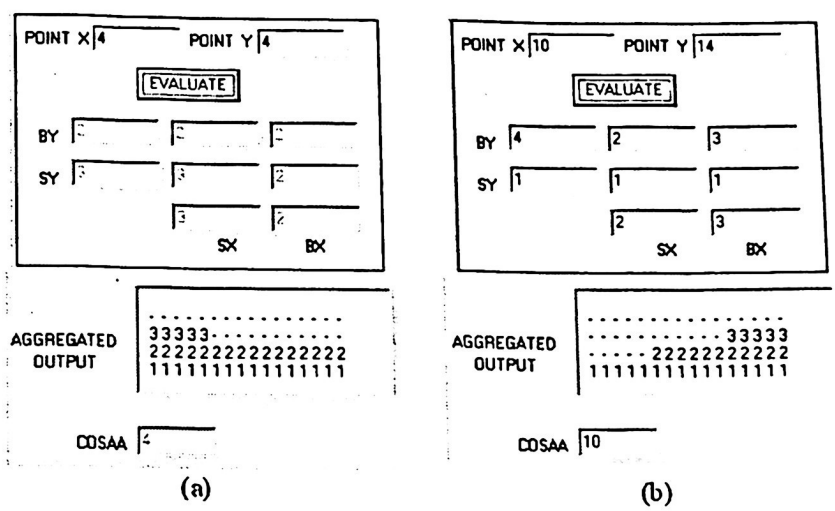

Fig. 9. Simulation Results for Two Different Cases, a) x=4, y=4. b) x=10, y=14

Simulator DiFES presented here was tested for different pairs of inputs *x, y*, where for each case a defuzzified crisp output was obtained using input and output membership functions as mentioned on previous sections. As can be seen on Fig. 9 a and b and on Fig. 10 a and b, here are four different cases, on top there are inputs for both points x and y, when evaluating this points, we obtain a crisp output at the bottom part of each figure, as result of COSAA defuzzifier. It can be easily seen that aggregated output membership function, changes its form according to the different cases presented.

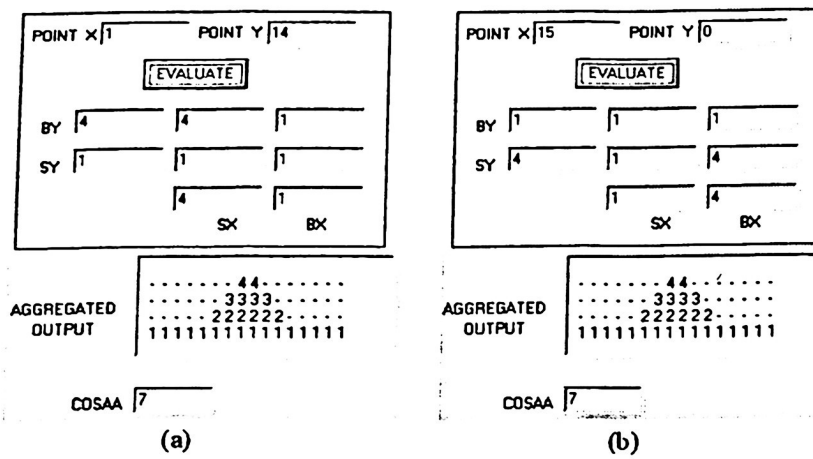

Fig. 10. Simulation Results for another Two Different Cases, a) x=1, y=14. b) x=15, y=0

5 Conclusions

In this work, there appeared the tool of simulation DiFES, which was used to demonstrate and to conclude that there is a form to represent fuzzy logic inference engine by means of simple operations such as additions and shifting, which gives as result a reduction on time consumed during complex calculation operands on conventional implementations [25][26][27][28], which use at less *n-1* iterations, our propose iteration number is less or equal than the *m* number of $\alpha - levels$ that is proportional to the number of bits used to discretize *y* axis into a truth space with height equal to 2^n-1.

The $\alpha - level$ method used to represent membership functions can be adapted to almost any shape, because it is represented as square slices with α height and a variable interval as its length and its resolution its on dependence on how many bits are used to discretize membership values.

Defuzzification method used calculates the center of area of every slice that forms aggregated output membership function, in terms of $\alpha - levels$ and calculate an average from them, requiring at most *m-1* iterations.

There justifies itself the need of the design of tools of simulation that allow us to experiment with new proposals of hybrid systems fuzzy-hardware.

6 References

1. Mamdani E. H., and Assilian S.: A case study on the application of fuzzy set Theory to Automatic Control. Proceedings of IFAC Stochastic Control Symposium. Budapest. (1974).
2. Sugeno M. and Murakami K.: An experimental study of fuzzy parking control using a model car. In: M. Sugeno, (eds.): Industrial Applications of Fuzzy Control. North-Holland, Amsterdam. (1985). 125-138.
3. Itoh 0., Gotoh K., Nakayama T., and Takamizawa S.: Application of fuzzy control to activated sludge process. Proceedings of 2ndIFSA Congress. Tokyo, Japan. (1987). 282-285.
4. Yagishita 0., Itoh 0., and Sugeno M.: Application of fuzzy reasoning to the water purification process. In M. Sugeno (eds.): Industrial Applications of Fuzzy Control, Amsterdam: North Holland. (1985). 19-40.
5. Kinoshita M., Fukuzaki T., Satoh T., and Miyake M.: An automatic operation method for control rods in BWR plants. In Specialists' Meeting on In-Core Instrumentation and ReactorCore Assessment. Cadarache, France. (1988).
6. Bernard J. A.: Use of rule-based system for process control. IEEE Control Systems Magazine Vol. 8, no. 5. (1988). 3-13.
7. Ruspini E. H.: Reactive Fuzzy Control of Autonomous Robots. On Hirota, Sugeno (eds.): Industrial Applications of Fuzzy technollogy in the world. World Scientific. (1995)
8. Ostrowski D. J. and Cheung P. Y. K.: A fuzzy Logic Approach to Handwriting Recognition. On M. j. Patyra, D. M. Mlynek (eds): Fuzzy Logic, Implementation and applications. Chapter 10. (1996)
9. Falchieri D., Gabrielli A., Gandolfi E., Masetti M.: Design of Very High Speed CMOS Fuzzy Processors for Applications in High Energy Physics Experiments. Seventh International Conference on Microelectronics for Neural, Fuzzy, and Bio-Inspired Systems. Granada, Spain. (1999). 284-291
10. Kosko B.: Fuzzy systems as universal aproximators. IEEE international conference on Fuzzy systems. Vol. 43 No. 11. (1992). 1329-1333.
11. Wang L. X.: Fuzzy systems are universal aproximators. IEEE Transactions on Systems Man and Cybernetics. (1992). 1163-1170.
12. Buckley: Sugeno type controllers are universal controllers. Fuzzy sets and systems. Vol. 53 No. 3. (1993). 299-303.
13. CastroJ. L.: Fuzzy logic controllers are universal aproximators. IEEE Transactions on Systems, Man and Cybernetics. (1995)
14. Togai M.: Expert system on a chip: an engine for real-time approximate reasoning. Proceedings of the ACM SIGART international symposium on Methodologies for intelligent systems. (1986). 147-154
15. Watanabe H.: RISC approach to design of fuzzy processor architecture. In Proceedings of 1st International Conference on Fuzzy Systems. (1992). 431–441
16. Cardarilli G.C., Re M., Lojacono R.: VLSI Implementation of a Real Time Fuzzy Processor. Journal of Intelligent and Fuzzy Systems. No. 3, Vol. 6. (1998). 389-401
17. Gabrielli A., Gandolfi E.: A Fast Digital Fuzzy Processor. IEEE Micro, Vol. 19, Issue 1, Jan.-Feb. (1999). 68-79.

18.Ascia G., Catania V., and Russo M.: VLSI Hardware Architecture for Complex Fuzzy Systems. IEEE Transactions on Fuzzy Systems. Vol.7, No.5. (1999). 553-570.
19. Surmann H., Ansgar Ungering and Karl Goser. Optimized Fuzzy Controller Architecture for Field Programmable Gate Arrays. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo. (1993). 124-133.
20. V. Salapura and V. Hamann. Implementing fuzzy control systems using VHDL and Statecharts. Proc. of the European Design Automation Conference EURO-DAC '96 with EURO-VHDL '96. Geneva, Switzerland. September. IEEE Computer Society Press. (1996). 53-58.
21. S. Sánchez-Solano, R. Senhadji, A. Cabrera, I. Baturone, C. J. Jiménez, A. Barriga. Prototyping of Fuzzy Logic–Based Controllers Using Standard FPGA Development Boards. Proc. 13th IEEE International Workshop on Rapid System Prototyping. Darmstadt. (2002). 25-32.
22. F.J. Garrigos-Guerrero y R. Ruiz Merino. Implementacion de sistemas fuzzy complejos sobre FPGAs. Computación reconfigurable y FPGAs. (2003). 351-358.
23. R. Raychev, A. Mtibaa, M. Abid. VHDL Modelling of a Fuzzy Co-processor Architecture. International Conference on Computer Systems and Technologies – CompSysTech (2005).
24. K. Uehara & M. Fujise, Fuzzy inference Based on Families of a-level Sets, IEEE Transactions on Fuzzy Systems, vol. 1, No. 2. May (1993). 111-124.
25. G.C. Cardarilli, M. Re, R. Lojacono, M. Salmeri, A new Architecture for High-Speed COG based Defuzzification. International Workshop on Tool Environments and Development Methods for Intelligent Systems, TOOLMET' 97, pp., April 17-18, (1997). 165-172.
26. Gaona, A.; Olea, D.; Melgarejo, M. Distributed arithmetic in the design of high speed hardware fuzzy inference systems;, 2003. NAFIPS 2003. 22nd International Conference of the North American Fuzzy Information Processing Society. 24-26 July (2003). 116 – 120.
27. Banaiyan, A.; Fakhraie, S.M.; Mahdiani, H.R.; Cost-Performance Co-Analysis in VLSI Implementation of Existing and New Defuzzification Methods. Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce. Volume 1. (2005). 828 – 833.
28. Hakaru Tamukoh, Keiichi Horio and Takeshi Yamakawa A bit-shifting-based fuzzy inference for self-organizing relationship (SOR) network IEICE Electronics Express, Vol. 4, No. 2. (2007). 60-65.