FPGA IMPLEMENTATION OF FUZZY CONTROLLERS

E. Lago, M. A. Hinojosa, C. J. Jiménez, A. Barriga, S. Sánchez-Solano

Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica
Avda. Reina Mercedes s/n, (Edif. CICA),
E-41012, Sevilla, Spain

XII Conference on Design of Circuits and Integrated Systems (DCIS'97), pp. 715-720, Sevilla, Noviembre 18-21. 1997

FPGA Implementation of Fuzzy Controllers

E. Lago, M. A. Hinojosa, C. J. Jiménez, A. Barriga, S. Sánchez-Solano

Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica
Edificio CICA, Avda. Reina Mercedes s/n, 41012-Sevilla
Phone: 95 4239923 - Fax: 95 4231832 - email: cjesus@imse.cnm.es

Abstract

This paper explores the use of FPGA technologies to implement fuzzy logic controllers (FLCs). Two different approaches are described. The first option is based on the logic synthesis of the boolean equations describing the controller input-output relations. The second approach uses dedicated hardware to implement the fuzzy algorithm according to a specific architecture based on a VHDL cell library. In both alternatives, the synthesis process is accelerated by means of CAD tools which translate a high level description of the controller. A set of design examples are included in order to analyze the application domains covered by the different solutions.

1. Introduction

The number of applications using fuzzy logic techniques to solve control and decision-making problems has increased considerably in the last years [1]. As a consequence, multiple solutions for the implementation of fuzzy algorithms resorting to either software and hardware approaches have been proposed and reported [2]-[17]. The applicability of these solutions depends on both the problem complexity (expressed as the number of input and output variables, and control rules) and the temporal restrictions (which establish the required inference speed). Software solutions provide flexibility to define the knowledge base, to select the fuzzy operators, and to choose the inference algorithms. However, they become inadequate for problems demanding high inference speed. In this case hardware solutions must be adopted [4].

Hardware realizations of fuzzy controllers can be accomplished following two general strategies. In the *off-line* strategy, output values are precomputed for all the possible input combinations and the inference process is carried out by a look-up table that can be implemented by storing the values in a RAM or by using combinational circuits [5]-[6]. Even though this approach allows for a complete flexibility in the definition of the fuzzy controller, its main drawback is the exponential growth of the required memory when the number of inputs or the number of elements in the universe of discourse is increased. To eliminate this handicap, the realization of many fuzzy controllers follows an *on-line* strategy by which dedicated hardware evaluates the inference process concurrently with the input changes [7].

Dedicated fuzzy hardware can be further classified in two categories: 1) *Fuzzy coprocessors*, which cooperate with a standard microprocessor to accelerate typical fuzzy operations as Min or defuzzification (Most of the commercially available fuzzy chips correspond to this category [8]-[11]); and 2) *Application Specific Fuzzy Hardware*, which implements the fuzzy algorithm in an integrated circuit [12]-[17]. A typical 'flexibility/speed' trade-off can be established between the two approximations. Fuzzy coprocessors provide a general purpose solution for high-complexity applications, while specific fuzzy hardware provides faster and cheaper implementations for middle and low-complexity problems.

This paper focuses on hardware implementations of low-complexity fuzzy controllers by means of Field Programmable Gate Arrays (FPGA). The choice of this implementation technology presents several advantages. First, it provides a fast prototyping capability for applications that can later be realized as integrated circuits. Second, systems built with FPGAs exhibit intrinsic programmability, thus providing a simple mechanism to change or adjust their function. Finally, a great number of development environments for FPGAs are commercially available.

Two different approaches for the realization of fuzzy controllers are considered (look-up table and specific fuzzy hardware). Both approaches are supported by design tools which bridge the gap between the high level specification of a controller and the representation required by FPGA synthesis tools. The applicability of the different solutions is analyzed by exploring the design space for some examples.

2. Design Flow for Fuzzy Controllers

The design flow of FLCs using our design tools is illustrated in Fig 1. The starting point of the synthesis process is a behavioral description of the controller using the specification language XFL [18]. The controller specification includes the initial knowledge base (antecedent and consequent membership functions, and rulebase) as well as the inference mechanism and the defuzzification method to be implemented. Such a description is suitable to be simulated and improved using the Xfuzzy 2.0 environment [19]. The refinement process is carried out with the help of a learning tool based on backpropagation algorithms. This tool allows us to identify the system rulebase and to adjust the membership function parameters.

When the controller specification is validated, the designer can choose two target implementations. The right branch in Fig 1 shows the steps to obtain a logic implementation based on boolean minimization. **Xftl** receives as input an XFL description and translates it to a look-up table with Berkeley's PLA format. The output PLA can be minimized and the boolean equations can be extracted by means of any compatible logic synthesis tool. In our case Xftl generates a *script* file for Synopsys to select the synthesis options.

The output of Synopsys is an XNF file (Xilinx Netlist Format) named '*fuzzy.sxnf*' with the controller description. Optionally it is possible to generate a report file containing routing requirements (number of CLBs and IOBs) and temporal constraints. Finally, *fuzzy.sxnf* is used as the input file for Xilinx software for mapping and routing the FPGA. Three files are obtained as result: '*fuzzy.lca*', '*fuzzy.bit*' and the

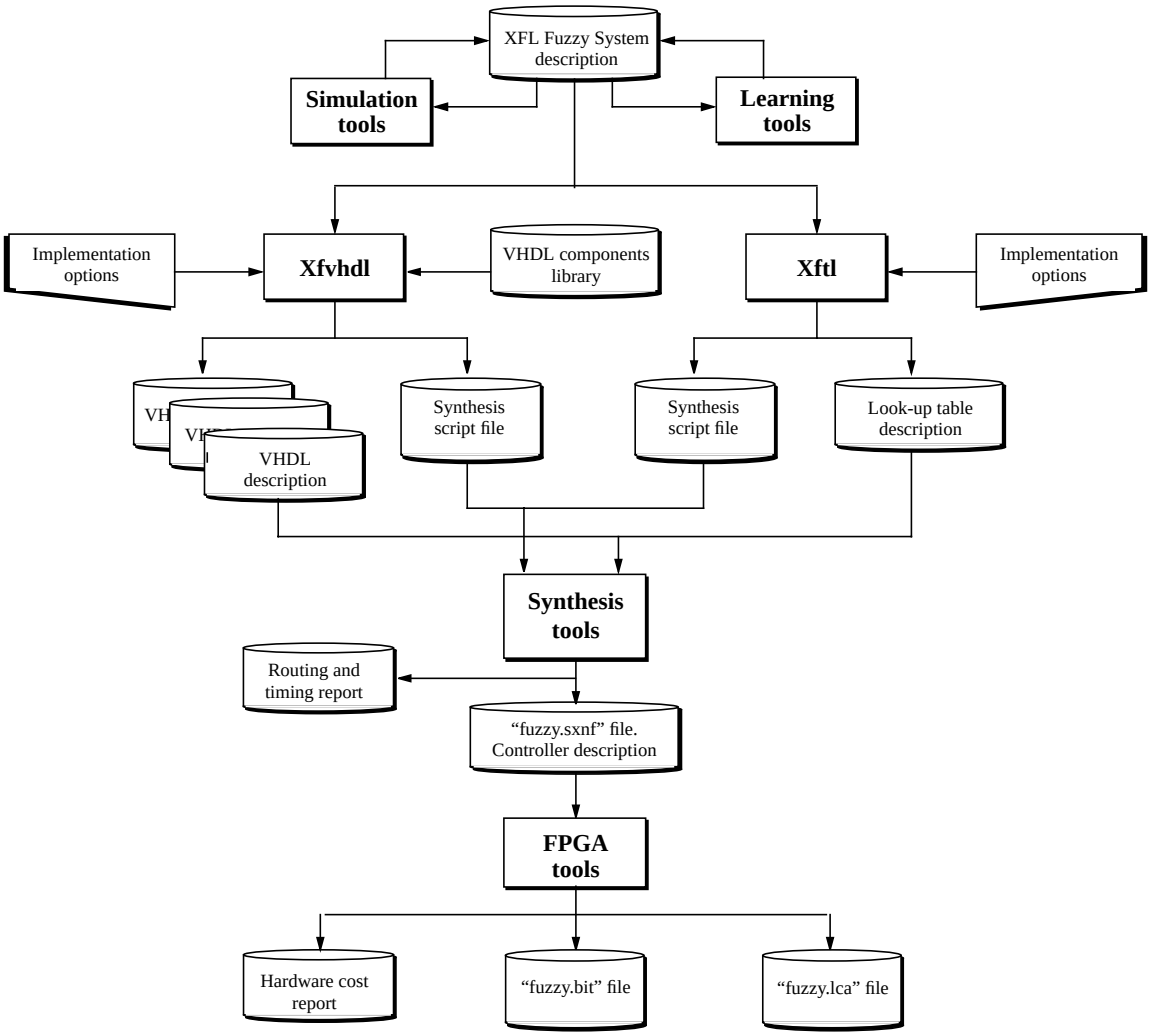


Fig. 1: Design flow for FPGA implementation of FLCs.

implementation report (in file '*fuzzy.rpt*'). The last step is to write the FPGA using the file '*fuzzy.bit*', to obtain the physical implementation of the fuzzy system from the behavioral XFL description.

An alternative implementation based on dedicated hardware can be accomplished by following the left branch in Fig 1. **Xfvhdl** reads an XFL file and generates a synthesizable VHDL description based on a specific architecture for fuzzy controllers (see [16] for a detailed description of this architecture). The architectural options and the number of bits of precision are defined by the user when the Xfvhdl command is run.

Xfvhdl uses a cell library containing the parameterized VHDL description for the basic building blocks. There are two kind of blocks: data path building blocks (implementing the inference algorithm) and control blocks (controlling the memory write/read operations and the signals that control the operation scheduling). The code used in the description of the cell library is compatible with the restricted VHDL implementations of the Synopsys and Mentor Graphics tools. Xfvhdl produces as output the following files describing the FLC:

Package files: Two VHDL packages are generated by Xfvhdl. The *constants* file includes the declaration of the constants used in the VHDL description. Some of these constants are obtained directly from the parameters of the Xfvhdl command. Others are obtained by analyzing the XFL description. The last set of parameters is calculated from the other two. On the other hand, the *entities* file contains the declaration of all the blocks that make up the FLC. The instantiation of each block depends on the architectural options selected.

Knowledge base files: The information about antecedents, rules and consequents is codified in a set of files. The definitions are based on tables of values by means of VHDL "case" sentences, thus enabling logic minimization when these blocks are implemented as combinational logic.

Controller file: This file contains the structural VHDL description of the FLC. The controller is constructed by concatenating a set of basic building blocks according to the XFL description and the implementation options.

TestBench file: In addition to the files required by the synthesis process, a testbench file is generated to ease the verification of the FLC. The testbench includes the instantiation of the FLC, a process which generates a periodical clock signal, and another process that provides the initial reset signal and the inputs used in the simulation of the FLC.

Xfvhdl also generates a *script* file to drive the Synopsys synthesis process. The steps required to obtain the FPGA are similar to the one previously described for the look-up table approach.

3. Design examples

In order to illustrate the versatility of the tools, and to analyze the application domains for the different implementation techniques, we will address in this section the realization of FLCs performing as function approximators. Two objective functions (Fig 2a and Fig 3a) will be studied [20]. We will explore the design space considering three implementation techniques (look-up table, and specific architecture using memory and arithmetic based MFCs [17]) and two defuzzification methods (Fuzzy Mean and Weighted

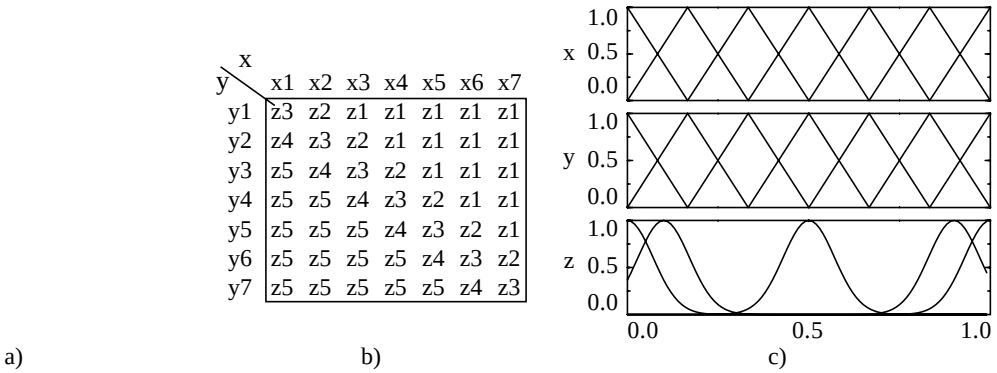
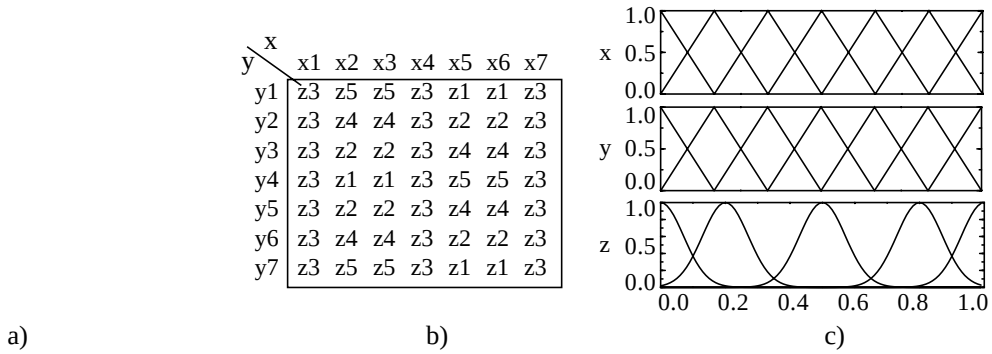


Fig. 2: Function F1: z=1/(1+exp(10*(x-y))). a) Function surface. b) Rule base.
c) Antecedent and consequent membership functions.



y \ x	x1	x2	x3	x4	x5	x6	x7
y1	z3	z5	z5	z3	z1	z1	z3
y2	z3	z4	z4	z3	z2	z2	z3
y3	z3	z2	z2	z3	z4	z4	z3
y4	z3	z1	z1	z3	z5	z5	z3
y5	z3	z2	z2	z3	z4	z4	z3
y6	z3	z4	z4	z3	z2	z2	z3
y7	z3	z5	z5	z3	z1	z1	z3

Fig. 3: Function F2: z=0.5(1+sin(2πx)cos(2πy)). a) Function surface. b) Rule base. c) Antecedent and consequent membership functions.

Fuzzy Mean). A precision range from four to fourteen bits will be analyzed for all the cases.

Using the Xfuzzy facilities we have identified the rulebases and adjusted the parameters for the resulting fuzzy controllers. Fig 2b and Fig 3b show the rulebases. Both examples use seven triangular membership functions to cover the input universes of discourse. Only five different values (z1, ..., z5) are used for consequents. Although consequents are represented in Fig 2c and Fig 3c as bell-shaped functions, only one or two significant parameters are required for simplified defuzzification methods which calculate the controller output as a weighted mean. For the Fuzzy Mean method (FM), the weight parameters are the bell-function centers. In the Weighted Fuzzy Mean (WFM), the second weight parameters correspond to bell-function widths.

The XFL descriptions were introduced as input files for the synthesis tools. All the controllers were designed using Xilinx XC4000 family FPGAs. To evaluate the performance of the different realizations, three characteristics have been considered: 1) implementation cost in terms of the number of CLBs; 2) approximation accuracy measured by the root mean square error (RMSE); and 3) inference speed in terms of main clock cycles. The results obtained are summarized in the next section.

4. Experimental Results

Fig 4a shows the evolution of the cost for the three implementation techniques when the number of bits is increased. For low resolution implementations the look-up table approach provides the best results. Conversely, as soon as the resolution grows (more than 5 bits), the specific hardware techniques address better results. In this sense, for higher resolution implementations, the arithmetic option seems to be more suitable than the memory option.

A similar analysis corresponding to the defuzzification methods is depicted in Fig 4b. Only the look-up table technique and the specific hardware with arithmetic MFCs have been considered for simplicity.

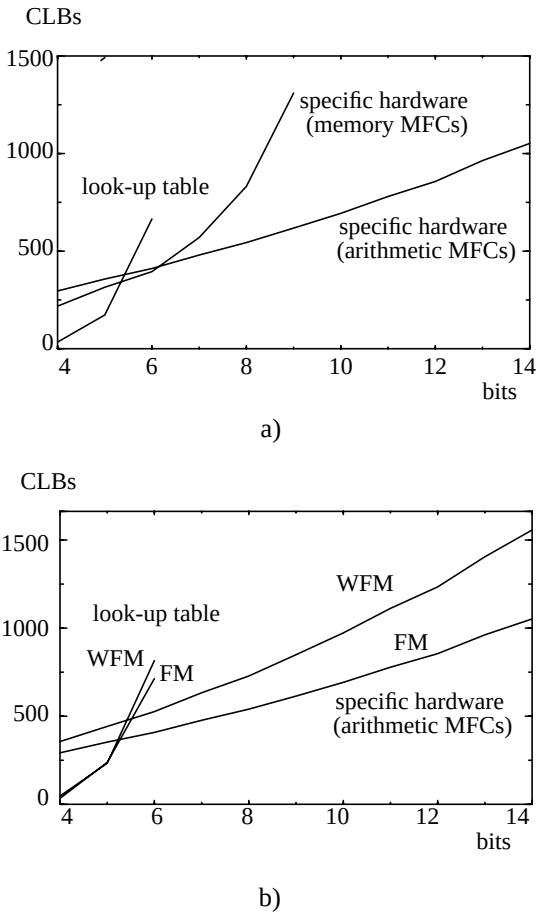


Fig. 4: Implementation cost comparison. a) Function F1 with FM, b) Function F2 with FM and WFM.

For look-up table techniques, the cost of WFM is slightly higher than that of FM. However, specific hardware implementation of WFM requires one additional multiplier, thus increasing the cost compared to FM.

Regarding approximation accuracy, errors can be caused by three causes: 1) the inherent fuzzy approximation error; 2) the truncation error due to limited bus width; and 3) the hardware implementation induced error. The first error is inherent to the approximated reasoning mechanism used by fuzzy logic systems. This error can be minimized by increasing the number of memberships for antecedents and consequents. To reduce the truncation error the bus width should be increased. Finally, the implementation error is a consecuence of the fixed-point arithmetic used in the controller. In certain parts of the circuit, such as in the defuzzification stage, this error is accumulative.

Fig 5a represents the RMSE as a function of the number of bits. The behavior is similar for the three implementation techniques. For low resolutions (up to 6 bits) the truncation error is the dominant factor. When resolution is increased, approximation and hardware errors become the main error sources. Comparing the three techniques, implementation errors have higher influence when specific hardware is used.

The results obtained for function F2 let us analyze how the defuzzification method influences the RMSE. As shows Fig 5b, WFM provides in general better results than FM. The difference between both defuzzification methods is more significant for the look-up table approach.

The final aspect of our study is focused on the evaluation of the inference speed provided by the different techniques. Implementations based on look-up tables are combinational circuits. In this case the inference speed is limited by the propagation delay of the FPGA critical path. The controller operation is slower when a specific architecture is used because several clock cycles are needed to calculate an inference. The system throughput depends on the bus width. On the other hand, latency is also fixed by the number of pipeline stages (two for memory based MFCs and three for arithmetic MFCs). Experimental results for five and six bits controllers implemented using the Xilinx 4013PQ160-5 FPGA prove that a clock frequency of 5.5 MHz. is sufficient to assure correct operation. This means an inference speed above 5 MFLIPS for look-up table controllers, and close to 1 MFLIPS for specific hardware controllers.

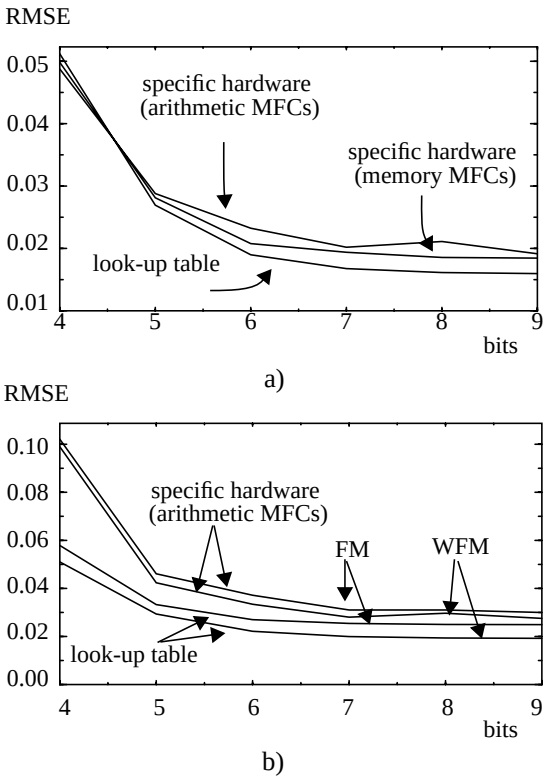


Fig. 5: RMSE comparison. a) Function F1 with FM. b) Function F2 with FM and WFM.

According to the reported results, a trade off between cost, accuracy and speed must be considered in the design process of a fuzzy controller. The look-up table technique is the best choice in terms of accuracy and speed, but cost criteria make this solution impracticable when the resolution grows. Techniques based on specific hardware are feasible for higher resolution controllers. Particularly architectures using arithmetic MFCs can be used to build high resolution controllers.

5. Conclusions

Two automatic synthesis tools for FPGA implementation of fuzzy controllers have been presented. These tools generate the controller circuit from a high-level behavioral description. One of the advantages is that (by definition of automatic synthesis tools) the implementation will be free of design errors. Other advantage is that the tools provide a mechanism for exploring the design space and let the designer choose the best solution. Xfuzzy 2.0 lets us do this with an additional advantage because it is an open environment. It permits us to introduce new elements (other controller architectures, other synthesis styles, etc).

References

[1] T. Munakata, Y. Jani, "Fuzzy Systems: An Overview", *Communications of the ACM*, vol. 37, n. 3, pp. 69-76, Mar. 1994.

[2] H. Surmann, A. P. Ungering. "Fuzzy Rule-Based Systems on General-Purpose Processors" *IEEE Micro*, vol. 15, n. 4, pp. 40-48, Aug. 1995.

[3] H. Watanabe. "A RISC approach to Design of Fuzzy Processor Architecture". *Proc. IEEE International Conference on Fuzzy Systems*, pp. 431-441, Mar. 1992.

[4] A. Costa, A. De Gloria, P. Faraboshi, A. Pagni, G. Rizzotto. "Hardware Solutions for Fuzzy Control". *Proceedings of the IEEE*, vol. 83, n. 3, pp. 422-434, Mar. 1995.

[5] J.Y. Leong, M.H. Lim and K.T. Lau, "A General Approach to Encoding Heuristics on Programmable Logic Devices", *Proc. fifth IFSA World Congress*, pp. 917-920, Seul, 1993.

[6] M. A. Manzoul and D. Jayabharathi, "FPGA for Fuzzy Controllers", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, n. 1, Jan. 1995.

[7] D. L. Hung. "Dedicated Digital Fuzzy Hardware". *IEEE Micro*, vol. 15, n. 4, pp. 31-39, Aug. 1995.

[8] K. Nakamura, N. Sakasmita, Y. Nitta, K. Shimomura, T. Tokuda. "Fuzzy Inference and Fuzzy Inference Processors". *IEEE Micro*, vol. 13, n. 5, pp. 37-48, Oct. 1993.

[9] M. J. Patyra, J. L. Grantner, K. Kirby. "Digital Fuzzy Logic Controllers: Design and Implementation". *IEEE Transactions on Fuzzy Systems*, vol. 4, n. 4, pp. 439-459, Nov. 1996.

[10] H. Eichfeld, T. Künemund, M. Menke. "A 12b General-Purpose Fuzzy Logic Controller". *IEEE Transactions on Fuzzy Systems*, vol. 4, n. 4, pp. 460-475, Nov. 1996.

[11] X. D. Kim, H. Lee-Kwang. "High Speed Flexible Fuzzy Hardware for Fuzzy Information Processing". *IEEE Transactions on Systems, Man and Cybernetics*, vol. 27, n. 1, pp. 45-56, Jan. 1997.

[12] H. Watanabe, W. Dettloff, K. Young, "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture", *IEEE Journal on Solid State Circuits*, vol. 25, n. 2, pp. 376-382, Apr. 1990.

[13] H. Ikeda, N. Kisu, Y. Hiramoto, S. Nakamura, "A Fuzzy Inference Coprocessor Using a Flexible Active-Rule-Driven Architecture", *Proc. IEEE International Conference on Fuzzy Systems*, pp. 537-544, San Diego, 1992.

[14] H. Eichfeld, M. Löhner, M. Müller, "Architecture of a CMOS Fuzzy Logic Controller with optimized memory organization and operator design" *Proc. IEEE International Conference on Fuzzy Systems*, pp. 1317-1323, San Diego, 1992.

[15] C. J. Jiménez, A. Barriga, S. Sánchez-Solano, "Digital Implementation of SISC Fuzzy Controllers", *Proc. International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pp. 651-652, Iizuka, 1994.

[16] C. J. Jiménez, S. Sánchez-Solano, A. Barriga, "Hardware Implementation of a General Purpose Fuzzy Controller", *Proc. IFSA World Congress*, vol. 2, pp. 185-188, Sao Paulo, 1995.

[17] S. Sánchez-Solano, A. Barriga, C. J. Jiménez, J. L. Huertas, "Design and Application of Digital Fuzzy Controllers", *Proc. sixth IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 869-874, Barcelona, 1997.

[18] D. R. López, S. Sánchez-Solano, A. Barriga: "XFL: a fuzzy logic systems language." *Proc. sixth IEEE International Conference on Fuzzy Systems*, vol. 3, pp. 1585-1591, Barcelona, 1997.

[19] A. Barriga, S. Sánchez-Solano, C.J. Jiménez, D. Galán and D.R. López: "Automatic Synthesis of Fuzzy Logic Controllers", *Mathware & Soft Computing*, Vol. III, n. 3, pp. 425-434. Sept. 1996.

[20] R. Rovatti and R. Guerrieri: "Fuzzy Sets of Rules for System Identification", *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 2, pp.89-102, May 1996 .