HARDWARE / SOFTWARE CODESIGN METHODOLOGY FOR FUZZY CONTROLLER IMPLEMENTATION

A. Cabrera[1], S. Sánchez-Solano[2], R. Senhadji[2], A. Barriga[2], C. J. Jiménez[2]

[1] Dpto. Automática y Computación. Facultad de Ingeniería Eléctrica. Instituto Superior Politécnico "José Antonio Echeverría", Calle 127 s/n, Marianao, Ciudad de la Habana, Cuba.
[2] Instituto de Microelectrónica de Sevilla, IMSE-CNM, Avda. Reina Mercedes s/n, 41012-Sevilla, Spain.

Abstract - This paper describes a HW/SW codesign methodology for the implementation of fuzzy controllers on a platform composed by a general-purpose microcontroller and specific processing elements implemented on FPGAs or ASICs. The different phases of the methodology, as well as the CAD tools used in each design stage, are presented, with emphasis on the fuzzy system development environment *Xfuzzy*. Also included is a practical application of the described methodology for the development of a fuzzy controller for a dosage system.

1. INTRODUCTION

Two opposite alternatives can be considered when exploring the design space of a complex electronic system. One of them is the use of standard components whose functionality can be defined by means of programming. The other one is the implementation of this functionality via a microelectronic circuit specifically tailored for that application. It is well known that the first alternative (the software alternative) provides solutions that present a great flexibility in spite of high area requirements and long execution times, while the second one (the hardware alternative) optimizes the size aspects and the operation speed but limits the flexibility of the solution. Halfway between both, hardware/software codesign techniques try to obtain an appropriate trade-off between the advantages and drawbacks of these two approaches.

Codesign techniques are also applicable to the development of fuzzy controllers. This paper presents a HW/SW codesign methodology for fuzzy controller implementation. The application of this methodology to a practical problem, the development of a fuzzy controller for a dosage system, is also described.

Section 2 shows some codesign alternatives for fuzzy controller implementation. Section 3 details each of the phases of the proposed methodology. A practical case for the application of the methodology is described in Section 4. Finally, Section 5 reviews the fundamental aspects of the present communication.

2. CODESIGN ALTERNATIVES

In a system codesign process, all the tasks that the system must carry out should be analyzed, evaluating the impact that the (possible) implementation options may have on the factors that define the overall system functionality and cost. The main parameters to consider in this evaluation are the task execution speed and the area required by its hardware implementation. Based on those results, a *partitioning* process is carried out, which consists in deciding which tasks should be executed by software and which should be implemented by hardware.

There are different approaches for applying codesign techniques in the development of control systems based on fuzzy logic. They are distinguished by the way in which the HW/SW partitioning is realized. One of them analyzes the influence of the instructions set of a processor on the implementation of a fuzzy inference system, redesigning it in such way that it supports those operations which best contribute to the increment of the inference speed [1]-[3]. In this case, the partitioning process is carried out at the instructions level of the processor.

Another alternative consists of implementing the inference system, totally or partially, by means of dedicated hardware with a specific architecture. This partition is based on the fact that, among all the tasks that a fuzzy logic-based control system must carry out, those related to the fuzzy inference calculation are the ones which consume the most time [2]. Therefore, their implementation by means of specific hardware will contribute significantly to increment the controller speed [4]-[6].

This codesign variant was the one chosen in this work. Whereas the remaining control tasks are programmed in software and executed by a standard microcontroller, the inference system is implemented on hardware following a specific architecture for fuzzy systems. This architecture is based on active rule processing, the use of membership functions with a maximum overlapping degree of two, and the employment of simplified defuzzification methods [6]-[8]. This architecture also presents different implementation options for fuzzification and defuzzification stages which must be considered in the codesign process.

3. CODESIGN METHODOLOGY

The different phases of the codesign methodology for fuzzy controller implementation are shown in Figure 1. Several CAD tools included in the fuzzy system development environment *Xfuzzy* [9] are required. This environment includes a group of tools which ease the different stages of description, verification and synthesis (software and hardware) of fuzzy systems. Next sections explain the main topics of each codesign phase, the CAD tools used, and the implementation details.

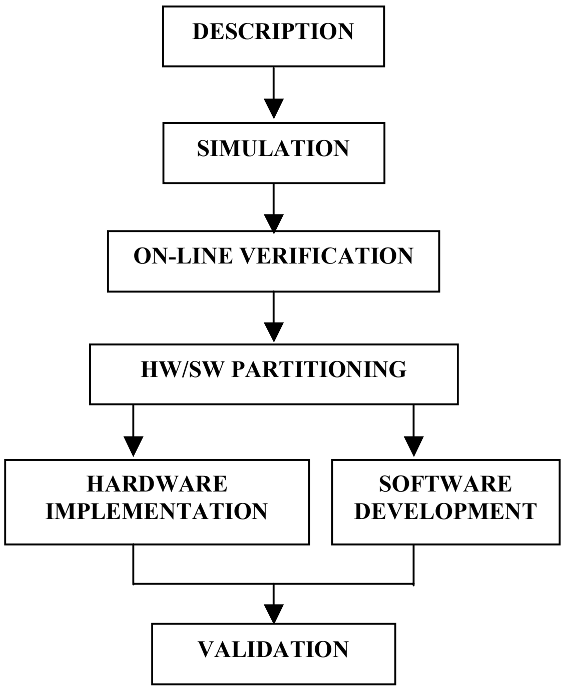


Figure 1. Codesign Methodology.

3.1. Description Phase

The description of the fuzzy inference system defining the controller operation (control strategy) is carried out during this phase using the fuzzy system specification language *XFL* [6], [10]. This language, used for all the tools included in *Xfuzzy,* allows us to define universes of discourse and membership functions for the controller input/output and to specify the system rulebase. It also permits the selection of the different fuzzy operators used as antecedent connectives, the implication function, the rule aggregation mechanism, and the defuzzification method. The *XFL* specification can be obtained by means of a conventional text editor or using the different editing options of the *Xfuzzy* graphical interface.

3.2. Simulation Phase

An *off-line* simulation of the behavior of the fuzzy inference system on the plant to be controlled is carried out in this phase. The main objective of this simulation is to obtain a preliminary adjustment of the control parameters. For this task, a model of the plant, described by means of C language, and a CAD simulation tool are required. The *xfsim* tool [6] provided by the *Xfuzzy* environment is used for this purpose.

Although this simulation phase allows a first adjustment of the inference system parameters, it should be noted that its results are only approximate ones due to the limitations of the model. Therefore, for a more precise adjustment of these parameters, interaction of the inference system with the real plant is required. This is the objective of the following phase.

3.3. On-Line Verification Phase

The main characteristic of this phase is the inclusion of the real plant within the control loop, in order to obtain the optimal adjustment of the fuzzy controller knowledge base. Obviously, as the inference system will finally be implemented on hardware, its parameters should satisfy the restrictions imposed by the architecture. For the same reason, the controller behavior should not only be evaluated with different knowledge bases but also with different resolutions (number of bits for parameter representation).

On the other hand, the interaction with the real plant requires more elements than those used in the simulation phase. The interface devices with sensors and actuators (signal amplifiers, A/D and D/A converters, etc.) as well as the pre- and post-processing routines (to obtain the controller input and output) are among them.

To develop this phase, a CAD tool is required which implements, through software, the fuzzy inference system as well as the pre- and post-processing routines, and allows the interaction with the real plant through a data acquisition card. The *Xfuzzy* development environment includes the *xflab* tool [11] for these purposes. *Xflab* allows the full development of a fuzzy controller implemented in software. With *xflab,* the model which represents the plant in the simulation phase is replaced by a function which monitors the real plant (through the data acquisition card), carries out the inference, and acts on the plant. Simultaneously, *xflab* registers the system variables which have been selected for their later processing with the objective of determining the best adjustment of the controller parameters. *Xflab* eases the data acquisition card configuration (input/output addresses setting, analog and digital channels selection, etc.), the information processing routines programming (linealization, filtering, etc.), and the setting of the control parameters (sampling rate, signals which must be recorded in order to measure the behavior of the controller, etc.).

Once the system configuration has been defined, *xflab* calls for another *Xfuzzy* tool, *xfc*, which obtains the C code of the inference system. *Xflab* then merges this code with the pre- and post-processing routines and with the monitoring routines. Finally, it compiles and links all these codes in order to obtain an executable file which corresponds to the software implementation of the fuzzy controller. Following this procedure it is very easy to modify the controller parameters (modifying the *XFL* specification) and to register the system performance. This way, different conditions can be tested in order to obtain the best approach for the system implementation.

Note that the *on-line* verification phase allows us to check and to adjust the global operation of the fuzzy controller. This is an important difference with respect to the simulation phase, which only allows a preliminary adjustment of the inference system. However, due to the difference of precision between software and hardware implementations, a later adjustment of the fuzzy controller parameters may be necessary. It would be done during the validation phase, the last one of the present methodology.

A fully functional software-implementation of the fuzzy controller has been obtained at the end of the verification phase. It constitutes the starting point of the following phase, the identification and partitioning of the system tasks.

3.4. Partitioning Phase

Although the specific tasks that a fuzzy controller should carry out will depend on the control application, they are usually related to the following general actions:

a) System initialization

b) Target determination

c) System sensor reading

d) Input pre-processing

e) Fuzzy inference

f) Output post-processing

g) Control signal generation

Concurrently with these tasks, the system timing should be included. This task will be in charge of setting down the sampling rate for the sensor signals. In addition, a mechanism for monitoring the controller variables should also be included. Registering the selected information in files would usually suffice.

Once the tasks which need to be executed have been identified, the HW/SW partitioning process can be realized. As explained in Section 2, the inference system will be implemented on hardware. The remaining tasks must be evaluated in order to obtain the best “speed vs. flexibility” trade-off. In general, the implementation of these tasks in software produces acceptable results. Finally, the implementation of other components (e.g. timing and monitoring mechanisms) must also be considered.

A fundamental aspect in this design step is the choice of the platforms where the tasks will be implemented, that is, the type of processor in which the software tasks will be executed and the hardware device which will support the inference system implementation. The availability and versatility of development boards which include a general-purpose microcontroller and an FPGA allow sharing between both components the functionality required by the fuzzy controllers. Thus, the calculation of the fuzzy inference will be assigned to a specific purpose controller built on the FPGA, while the pre- and post-processing algorithms, as well as the rest of the tasks, will be programmed in the microcontroller. The timing process will be implemented using one of the timers available in the microcontroller. To complete the controller it will be necessary to add the circuits (signal amplifiers, A/D and D/A converters, etc.) in order to adapt the signal levels used by the sensors and actuators to those required by the development board.

3.5. Hardware Implementation Phase

A CAD tool allowing the translation of the inference system specification to a format which can be synthesized (a hardware description language) and supports the architecture described in [6]-[8] is required for the development of this phase. The main advantage of the *Xfuzzy* development environment when compared with other similar environments is the inclusion of tools that support the hardware synthesis of the inference system. One of these tools is *xfvhdl* [12], which allows us to obtain a VHDL description of the inference system from its *XFL* specification. This VHDL description is compatible with different hardware synthesis tools for FPGAs or ASICs. The use of FPGAs is especially adequate in order to obtain a first prototype of the system.

Note that the FPGA must not only contain the inference system, but also the input/output interface elements with the microcontroller and other elements which might be necessary due to the characteristics of the development board or the controlled system.

3.6. Software Development Phase

During this phase, different routines are developed which support the tasks that will be implemented by the software. Therefore, program development and debugger tools (assemblers or compilers, simulators, etc.) are used.

Although it is not necessary, it may be advisable to develop a set of programs designed to diagnose and check the operation of the different parts of the controller.

3.7. Validation Phase

The integration of hardware and software implementations on the development board and the definitive adjustment of the controller parameters are carried out during this phase. Once the operation of the different parts has been debugged, an integral validation of the system should be carried out. The implemented fuzzy controller is put into operation and the group of selected variables (used for evaluating the system behavior) is transmitted to a personal computer. Then, with the help of information processing routines, important data (settling time, overshoot, rise time, mean quadratic error, etc.) can be obtained in order to characterize the implemented fuzzy controller. Since microcontroller tasks and inference system parameters can be easily modified, different versions of the controller can be validated in order to obtain the one with the best performance.

4. APPLICATION OF THE METHODOLOGY

In order to illustrate the described codesign methodology, the development of a fuzzy controller for a real system is presented in this section.

4.1. System Description

The system to be controlled consists of a scaled model of a dosage system designed to experiment with different control strategies based on fuzzy logic (Figure 2).

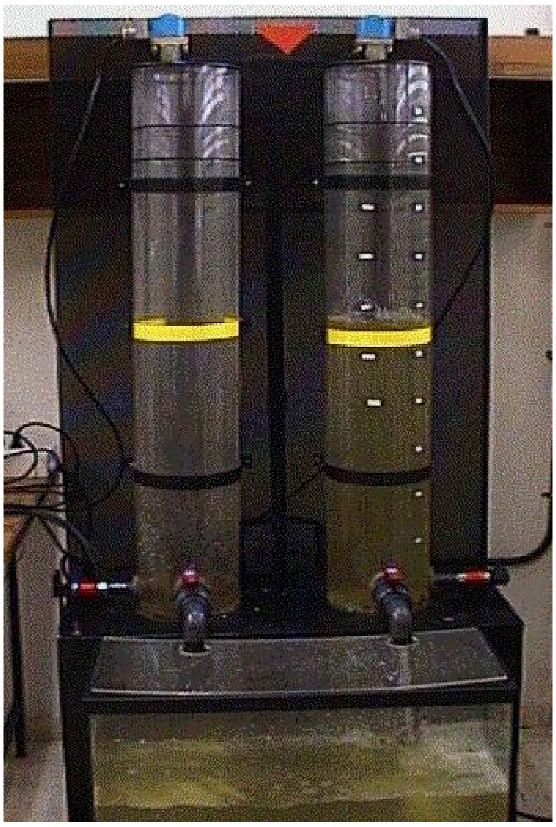


Figure 2. Dosage system provided by SUR A&C.

An electrically controlled water pump and two cylindrical tanks (120 cm high and 20 cm wide), each one with an electronic valve for liquid injection and a manually controlled output valve for liquid discharge, compose the system. The liquid level is obtained through a pressure sensor located at the bottom of the tank. The water pump and the injection valves are voltage-controlled devices (0 – 10 volt) whereas the pressure sensors have a current signal output (4 – 20 mA). However, the latter devices are industrial ones which work in a greater range, so their real output in this application is only between 4 and 5,6 mA.

Three different types of level-controlled-systems can be tested with this plant. Two of them are composed by only one tank and the difference between both is the control element: the electronic valve or the water pump. In both cases, the other device (the pump or electronic valve) remains in a fixed position. These one-tank systems are functionally equivalent to fuzzy control actions based on equivalent rules. Therefore, the same fuzzy controller can be applied to both one-tank systems.

The third type uses two tanks and the controller must act on the motor pump and both electronic valves at the same time. The goal in the three systems is to maintain the level in each tank as near as possible to the target control positions, with the lowest possible overshoot, and with the smallest transient time.

The applied control strategy corresponds to the fuzzy equivalent of an incremental PI controller. Therefore, the fuzzy controller for one-tank systems has two inputs (error and error variation) and its output is the change of the valve aperture or the change of the pump motor speed.

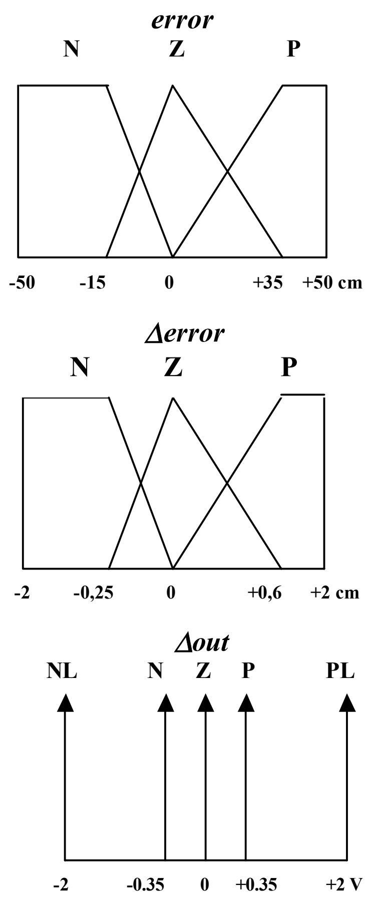
Two independent fuzzy controllers can be implemented for the whole plant, each one acting upon one electronic valve, whereas the pump will be controlled by the bounded sum of the output of both controllers. Nevertheless, due to the symmetry between the tanks, both fuzzy controllers will be identical. Thus, only one will be necessary if the inference processes are executed sequentially.

4.2. Implementation Details

Using the present methodology, the *XFL* description phase was developed for an inference system corresponding to an incremental fuzzy-PI controller for a one-tank system. The input (error and error variation) and output (variation of the voltage applied to the electro-valve) membership functions were estimated based on the behavior of the system, but also considering the limitations imposed by the future hardware implementation. For this reason, three membership functions with overlapping degree of two were used for the inputs, five singletons for the output, and defuzzification by the Fuzzy Mean method. Figure 3 shows these membership functions as well as the system rule base.

Afterwards, the system behavior was simulated by means of *xfsim* using a simplified model of the plant. This simulation was useful for a first adjustment of the inference system knowledge base.

A better adjustment was obtained during the *on-line* verification phase when the controller was inserted into the control



Δerror \ *error*	N	Z	P
N	Z	P	PL
Z	N	Z	P
P	NL	N	Z

Figure 3. Input / Output membership functions and rule base

loop. Using *xflab,* 27 different implementations (in which the input/output memberships functions were modified) were carried out. In each implementation, the system works during 2000 iterations using four different target positions. A group of selected variables (error, error variation, liquid level, output variation, etc) was registered for each test and was later processed obtaining the data (rise time, overshoot, mean quadratic error, etc.) which allows us to select the *XFL* specification with the best performance [13].

Based on the previous results, the fuzzy controller for the two-tank system was also developed with *xflab*, obtaining quite similar behavior. Thus, the *XFL* specification which must later be implemented on hardware was obtained. In addition, all the tasks which the fuzzy controller must carry out were identified. This was the starting point for the HW/SW partitioning stage.

The platform used to implement the controller was the development board XS-Board (XS40-005XL, from XESS Corporation [14]), which has an Intel 8031 microcontroller, a Xilinx XC-4005 FPGA, 32 Kbytes of static RAM, a programmable clock circuit, and other devices. XESS provides software facilities which allow for the setting of the clock circuit, configuring the FPGA, and downloading the microcontroller program.

In order to minimize the resources of the available FPGA, the symmetry between both one-tank controllers was considered. Therefore, only one controller was implemented on the FPGA and two consecutive inference cycles were executed.

In the hardware implementation phase, the FPGA synthesis was carried out with the “Xilinx Foundation 3.1i” environment [15]. The VHDL description of the inference system was obtained based on its *XFL* specification and using the *xfvhdl* tool. Controller input/output resolutions were limited to 6 bits. The controller uses memory–based antecedents and the Fuzzy Mean defuzzification method. On the other hand, since the inference system acts as an 8031 coprocessor, it was also necessary to incorporate interface circuits with this device in the FPGA, as well as the circuitry necessary to access the memory. The result of the implementation led to a 98% of CLBs used in the FPGA (193 of 196) and a maximum frequency of 20 MHz. At 10 MHz (for both the FPGA and the microcontroller) the hardware inference cycle delays 700 nanoseconds, which is lower than the 8031 single instruction execution time. The developed experimental assembly consisted of an XS-Board inserted into another board which also includes the A/D (AD7824) and D/A (AD7226) conversion devices and the signal conditioning circuits.

Software implementation on the 8031 was carried out with the aid of available development tools for the MCS-51 family. The microcontroller program includes the general algorithm for the control cycle (input acquisition, preprocessing, communication with the inference engine, postprocessing, and output generation), together with several auxiliary routines which control the interrupt mechanisms and the communication with a PC.

A first implementation of the fuzzy controller was carried out based on the same *XFL* specification which led to the best behavior in the verification phase. The obtained results, although acceptable, were far from the expected ones due to the precision difference between software and hardware inference. Therefore, it was necessary to realize additional adjustments of the memberships functions shown in Figure 3. Hardware implementation was carried out again and the controller parameters were processed. The behavior was better than in the previous one but not as good as the one from the verification phase. The sampling time was then reduced from 500 to 100 ms (by means of a simple modification in the microcontroller program) obtaining very similar results to those obtained with *xflab* control. Figure 4, representing the evolution of the level of liquid in a tank for different target positions, shows some of the experimental results.

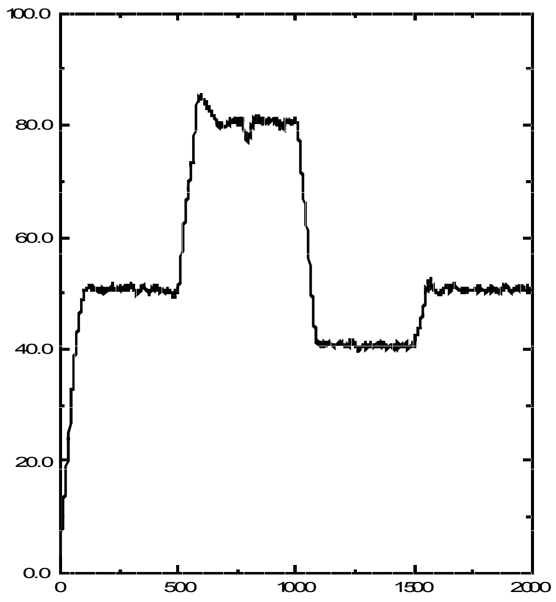


Figure 4. Liquid level vs. samples in tank-2 (each sample corresponds to 500 ms).

5. SUMMARY

A codesign methodology for fuzzy controllers, based on the *'a priori'* partition of the tasks implemented by hardware and software components, has been presented. The *Xfuzzy* environment integrates all the necessary tools to carry out the design flow of a fuzzy logic-based system (from the high-level description to its implementation according to this methodology). Finally, the application of the methodology to a real problem was discussed. A very attractive option, for its combination of speed and flexibility, consists of implementing the fuzzy controller by using commercial development boards which include microcontrollers and FPGAs.

REFERENCES

[1] Von Altrock, C., "*Adapting existing Hardware for Fuzzy Computation*", Institute of Physics Publishing, 1998.

[2] Salapura, V., "A Fuzzy RISC Processor", *IEEE Transactions on Fuzzy Systems*, Vol. 8, N. 6, 2000.

[3] Ungering, A.P., Bauer, H., Goer, K., "Architecture of a Fuzzy Processor Based on an 8-bit Microprocessor", *Proc. IEEE ICFS'94*, Orlando, pp. 297-301, 1994.

[4] Costa, A., Da Gloria, A., Faraboschi, P., Pagni, A., Rizzotto, G., "Hardware Solutions for Fuzzy Control", *Proceedings of the IEEE*, Vol. 83, N. 3, pp. 422-434, 1995.

[5] Hollstein, T., Kirschbaum, S., Halgamuge, S., Glesner, M., "Application Specific Fuzzy Processor", *Proc. Int. Symposium on Nonlinear Theory and its Applications*, pp.133-136, Honolulu, 1997.

[6] Baturone, I., Barriga, A., Sánchez-Solano, S., Jiménez, C.J., López, D., "*Microelectronic Design of Fuzzy Logic-Based Systems*", CRC Press, 2000.

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

[8] Jiménez, C.J., Sánchez-Solano S., Barriga, A., "Hardware Implementations of a General Purpose Fuzzy Controller", *Proc. Sixth International Fuzzy Systems Association World Congress*, Vol. 2, pp.185-188, Sao Paulo, 1995.

[9] López, D.R., Jiménez, C.J., Baturone, I., Barriga, A. Sánchez-Solano, S., "Xfuzzy: A Design Environment for Fuzzy Systems", *Proc Seventh IEEE International Conference on Fuzzy Systems*, pp. 1060-1065, Anchorage, 1998.

[10] López, D.R., Moreno, F.J., Barriga, A. Sánchez-Solano, S., "XFL: A Language for the Definition of Fuzzy Systems", *Proc. Sixth IEEE International Conference on Fuzzy Systems*, Vol. 3, pp. 1581-1591, Barcelona, 1997.

[11] Senhadji, R., Sánchez-Solano, S., López, D.R. Barriga, A., "Xflab: An On-line Verification Tool for Fuzzy Controllers", *Proc. International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems*, Vol. 1, pp. 44-49, Madrid, 2000.

[12] Lago, E., Jiménez C.J., López D.R., Sánchez-Solano S., Barriga A., "XFVHDL: A Tool for the Synthesis of Fuzzy Logic Controllers", *Proc. Design, Automation and Test in Europe*, pp. 102-107, Paris, 1998.

[13] Cabrera, A., Senhadji, R., Sánchez-Solano, S., Barriga, A., Jiménez, C.J., Llanes, O., "Development of Level Controllers Based on Fuzzy Logic", *Proc. First International ICSC-NAISO Congress on Neuro Fuzzy Technologies,* Ciudad de la Habana, 2002.

[14] *XS40, XSP Board V1.4 User Manual*, XESS Corporation, USA, 1999.

[15] *The Programmable Logic Data Book*, Xilinx Inc., USA, 1999.