CAD Tools for Hardware Implementation of Embedded Fuzzy Systems on FPGAs

María Brox, Santiago Sánchez-Solano, Ernesto del Toro, Piedad Brox, and Francisco J. Moreno-Velo

***Abstract*—This paper describes two computer aided design (CAD) tools for automatic synthesis of fuzzy logic-based inference systems. The tools share a common architecture for efficient hardware implementation of fuzzy modules, but are based on two different design strategies. One of them is focused on the generation of standard VHDL code, which can be later implemented on a reconfigurable device (FPGA) or as an application specific integrated circuit (ASIC). The other one uses the Matlab/Simulink environment and tools for development of digital signal processing (DSP) systems on Xilinx´s FPGAs. Both tools are included in the last version of *Xfuzzy*, a specific environment for designing complex fuzzy systems, and they provide interfaces to commercial VHDL synthesis and verification tools, as well as to conventional FPGA development environments. As demonstrated by the included design example, the proposed development strategies speed up the stages of description, synthesis, and functional verification of embedded fuzzy inference systems.**

***Index Terms*—CAD tools, Fuzzy inference systems, FPGAs, Hardware implementation.**

I. Introduction

Fuzzy logic provides an adequate tool to deal with the uncertainty and imprecision typical of the reasoning system used by the human brain [1]. In particular, the capability of fuzzy systems to capture the knowledge of human experts and translate it into robust control strategies by means of IF-THEN rules similar to those employed in natural language (without the need of mathematical models) has motivated a considerable increase in the number of control applications using techniques based on fuzzy inference in the last years [2]-[4]. Many different approaches for hardware implementation of fuzzy systems by means of ASICs or FPGAs have been also proposed in the literature [5]-[7].

The recent improvements in FPGA technologies has led to important advances in programmable logic devices, which allow the implementation of a complete system on a programmable chip (SoPC) [8]-[10]. In addition to the classical VHDL-based design flow [11]-[13], FPGA manufactures have recently developed different design tools, such as System Generator from Xilinx (XSG) [14], to ease the implementation of digital signal processing (DSP) algorithms on FPGAs [15]-[17]. Hardware implementation of fuzzy inference systems is currently demanded by different applications of industrial control, robotics or consumer electronics [18]-[25]. However, regarding the application of computational intelligence paradigms, to benefit from these technological advances, it is necessary the development of powerful CAD tools that allow automating the different design stages of a fuzzy inference system and translating its high-level description into an efficient hardware implementation. The use of adequate CAD tools reduces the development cycle of new products and makes them more competitive in market terms.

The huge success of fuzzy logic in the last decade of the past century caused the development of many tools dedicated to the design of fuzzy inference systems. Most of these tools were focused on the acquisition of knowledge, i.e., the creation of fuzzy systems from a data set, the tuning of system parameters using learning algorithms, and the comparison of different fuzzy operators and inference approaches [26]-[29]. Many of the initially available commercial software (such as TIL Shell from Togai InfraLogic, FIDE from Aptronix, and FuzzyTECH from Inform) allowed generating optimized code for the different families of microcontrollers existing in that epoch. Different proposals for automatic synthesis of fuzzy systems using specific hardware based on analog and digital design techniques were also presented in those years [30]-[33]. Digital approaches were usually based on the generation of general-purpose or specific VHDL code. More recently, a great number of different proposals use the facilities included in the Matlab/Simulink environment for the design and hardware implementation of fuzzy systems on reconfigurable devices [34]-[38].

An interesting CAD tool that offers facilities for the whole development cycle of a fuzzy system is the *Xfuzzy* environment [39]. Two hardware synthesis tools currently integrated into *Xfuzzy* are described in the paper. One of them employs a revised strategy based on VHDL, which generates code that can be synthesized and implemented on ASICs or

M. Brox is with the Department of Computer Architecture, University of Córdoba, Córdoba, Spain (phone: +34 957212224; e-mail: mbrox@uco.es).

S. Sánchez-Solano is with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain (e-mail: santiago@imse-cnm.csic.es).

E. del Toro is with the Microelectronics Research Center (CIME-CUJAE), Havana, Cuba (e-mail: ernesto@electrica.cujae.edu.cu).

P. Brox is with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain (e-mail: brox@imse-cnm.csic.es).

F. J. Moreno-Velo is with the Department of Applied Physics and Electrical Engineering, University of Huelva, Huelva, Spain (email: francisco.moreno@dti.uhu.es).

FPGAs. The other one provides FPGA implementations based on XSG. It takes the advantages of flexibility and ease of configuration offered by Matlab.

The structure of the paper is the following. Both design strategies are based on an efficient hardware architecture for fuzzy systems, which is described in Section II. The first synthesis tool, as well as the VHDL cell library supporting it, is detailed in Section III. The second tool and its associated cell library developed in Simulink are presented in Section IV. Section V illustrates a design example where a hierarchical fuzzy system has been generated from a set of numerical data by means of the identification facilities provided by *Xfuzzy*. The design flow and the supporting tools associated to both design strategies are detailed in this section. Finally, Section VI summarizes the main conclusions of this work.

II. Active-rule Based Architecture for Fuzzy Systems

From an implementation point of view, a fuzzy system is composed by three stages: fuzzification, inference, and defuzzification. The fuzzification stage is in charge of accepting the inputs to the inference system and evaluating the similarity degree between these inputs and the membership functions associated to the linguistic labels used in the rule antecedents. The inference engine evaluates the different rules in the knowledge base. The activation degree of each rule is calculated from the activation degree of its antecedents and according to the interpretation of the different connectives in use. Finally, the conclusions of the different rules are combined and the defuzzification stage is used to provide the output of the inference system. The implementation scheme followed by both automatic synthesis tools is based on the architecture shown in Fig. 1. This architecture allows efficient implementations of digital fuzzy systems in terms of use of resources and inference speed. Its main characteristics are the limitation of the overlapping degree of input membership functions, a processing strategy that evaluates only the active rules, and the use of simplified defuzzification methods [5], [22].

The block diagram of this architecture illustrates the three stages needed for the calculation of fuzzy inference based on Mamdani's model. In the fuzzification stage, the membership functions circuits (MFC) provide as many pairs "label, activation degree" (L_i, μ_i) for each input value as overlapping degree has been fixed for the system. The different combinations of these labels will determine the possible rules that are activated. In the following stage, the inference process is carried out by sequentially processing the active rules by means of an active-rule selection circuit composed by a counter-controlled multiplexer array. In each clock cycle, the membership degrees (μ_i) from rule antecedents are combined within the connective-antecedent operator circuit to calculate the activation degree of the rule (h_i), while the antecedent labels address the rule memory location containing the parameters that define its corresponding consequent. The number and meaning of these parameters depend on the defuzzification method that is being employed [5] (c_i, w_i, for Mamdani's systems, and a_i, b_i, c_i, for Takagi-Sugeno's systems). In the defuzzification stage, a crisp output value is

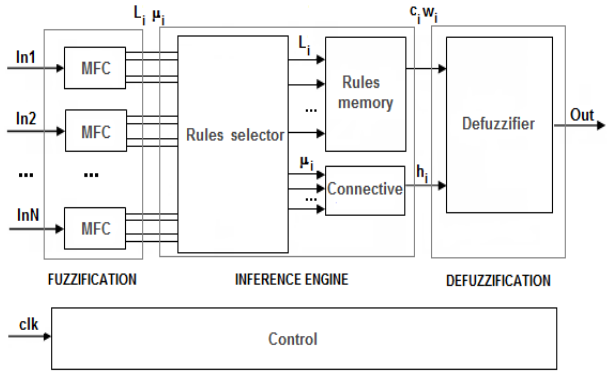


Fig. 1. Architecture for a Mamdani's type fuzzy system with a processing strategy based on active rules

obtained by processing the rule consequents with its different activation degrees, according to the selected defuzzification method. This stage is carried out in two processes, one that is in charge of the accumulation tasks and another that performs the division (if required), according to (1) for Mamdani's or (2) for first-order Takagi-Sugeno's systems. For Fuzzy Mean (where $w_i = 1$) and Takagi-Sugeno methods, the division operation can be avoided when $\sum h_i = 1$ (this condition is always fulfilled for 2-input inference systems using triangular membership functions with an overlapping degree equal to two and the product operator as antecedent connective).

$$\bar{y}_M = \frac{\sum_i h_i \cdot c_i \cdot w_i}{\sum_i h_i \cdot w_i} \quad (1)$$

$$\bar{y}_{TSK} = \frac{\sum_i h_i \cdot (x_1 \cdot a_i + x_2 \cdot b_i + c_i)}{\sum_i h_i} \quad (2)$$

A control block in charge of controlling the operation of the rest of blocks has been also included in the diagram.

Regarding timing considerations, a different pipeline stage may be introduced for each of the three stages in the architecture shown in Fig. 1. The minimum number of clock cycles required to perform the tasks associated with these pipeline stages will be the greatest of: the maximum number of membership functions, the number of active rules, and the number of bits of the output. As an exception, systems that use specific defuzzification methods where the division process can be avoided require fewer pipeline cycles, since they only perform the accumulation process in the last stage. In this case, the number of clock cycles required for the system operation will only be the greatest of the maximum number of membership functions and the number of active rules.

This architecture is characterized by being highly configurable due to the availability of different circuit realizations for the blocks in Fig. 1 [22]. MFCs can be implemented using either memory- or arithmetic-based approaches. Memory-based MFCs employ memory blocks to store the labels and membership degrees corresponding to each point of the input universe of discourse, thus allowing the use of fuzzy sets with unrestricted shapes. Alternatively, arithmetic-based MFCs employ arithmetic circuits to generate families of membership functions with triangular shapes (except the first and the last functions which can be of type "Z" and "S", respectively). An antecedent memory block for

each input stores, in this case, the values of "point, slope" (a_n, p_n) defining their associated membership functions. Using these values, the membership degree corresponding to a given input is calculated by an arithmetic block that performs the following operation when $input > a_n$:

$$\mu_n = (input - a_n) \cdot p_n \quad (3)$$

The designer can also choose between two options (product or minimum) for the connective used in the knowledge base of the inference system. Finally, depending on the kind of fuzzy module being implemented, different defuzzifier blocks can be selected: FuzzyMean, WeightedFuzzyMean, or first-order Takagi-Sugeno, for interpolators; and MaxLabel, for decision-making systems.

Regarding arithmetic considerations, fixed-point arithmetic is used by both design strategies. Input and output signals are normalized in the interval [0, 1]. The user can select the word-lengths of input and output signals, membership degrees, slopes, and defuzzification weight factors (w_i) using the graphical interfaces described in the following sections. The position of the binary point of slope values is automatically calculated in order to achieve the higher precision possible for the selected word-lengths. Results of internal operations are truncated according to the number of bits used by the output signals. Finally, when division is required, a serial non-restoring algorithm that only requires addition, subtraction and shifts operators is performed.

III. Hardware Synthesis with XFVHDL

In order to automate the design process with the proposed architecture, the VHDL-based technique has required the generation of a library of configurable and synthesizable blocks described in VHDL language. The blocks of this library implement the components of the architecture discussed in the previous section. Different blocks have been designed for the generation of membership functions, the sequential processing of active rules, the calculation of the rule activation degree, the storage of consequents for the rulebases, the development of tasks of accumulation and division for different defuzzification methods, and the generation of control signals. The library also contains a set of crisp blocks that implement general purpose arithmetic, such as addition, subtraction, multiplication or division functions, and logic operations, as a selector.

Building a fuzzy system with this library implies choosing and interconnecting the adequate elements, as well as assigning values to their parameters. VHDL descriptions of library components are parameterized by "generic" VHDL statements (generic parameters that are fixed when the component is placed), which facilitates the design process automation. VHDL descriptions of these blocks verify the restrictions imposed by the main synthesis tools, so they can be used as input to ASIC and FPGA design environments in order to allow rapid development of prototypes built to validate new ideas or evaluate possible solutions.

In order to speed up the design process based in this strategy, a new version of the *xfvhdl* synthesis tool has been recently incorporated into the *Xfuzzy* environment. This tool allows translating the high-level description of a fuzzy system written in XFL3 (the specification language shared by all the tools in *Xfuzzy*) into a VHDL description that can be synthesized and implemented on a programmable device or an application specific integrated circuit. The Graphical User Interface (GUI) of the new version of *xfvhdl* is shown in Fig. 2. The middle left area represents the knowledge base, structured as a pull-down menu with components grouped

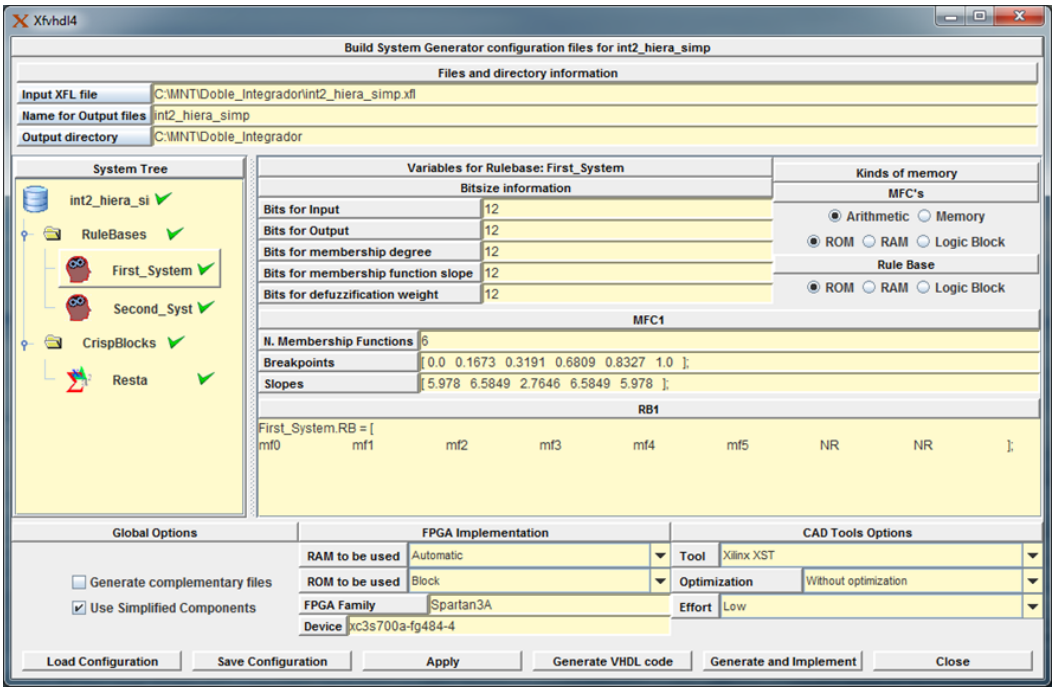


Fig. 2. Graphical User Interface of *xfvhdl*

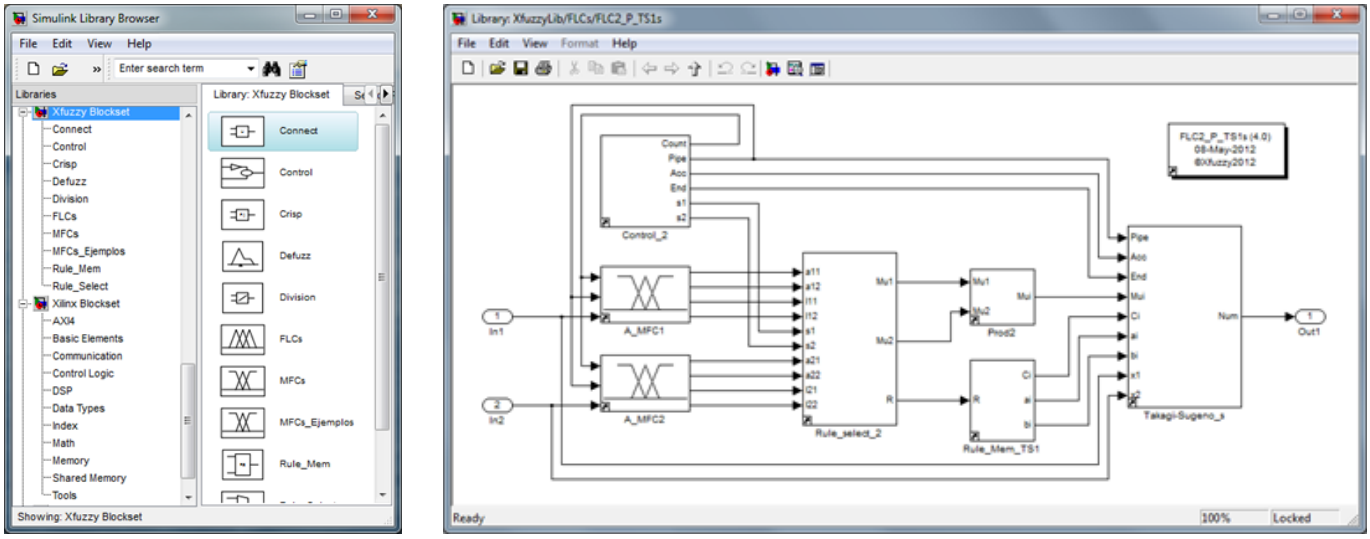
under the categories “RuleBases” and “CrispBlocks”. When a particular rulebase is selected, the middle right area allows defining parameters related to the dimension of the system. This area also shows information about membership functions and fuzzy rules extracted from the XFL3 specification. The user can select the implementation option for the antecedents: arithmetic- or memory-based. FPGA development tools usually allow selecting the type of memory used in the implementation stage. For this reason, the user can also choose the kind of memory (RAM, ROM or logical block) that will be used in MFC and rule memory blocks.

When all the architectural options have been selected and the parameters related to the bus sizes have been defined, the fuzzy system components are identified by green marks closed to them. Then it is possible to generate the output files by pressing the button “Generate VHDL code”. Basically, this action generates a VHDL description of the fuzzy system and a testbench, also described in VHDL, that allows verifying the functionality of the system. The VHDL file is based on the interconnection of the library blocks that have been described in the previous section. At the beginning of this file a package of constants is introduced. These constants are automatically calculated from the values extracted from the knowledge base and the parameters related to the bus sizes that are introduced by the user. When the interconnection of library blocks is performed, the generic parameters of each block are connected to the constants defined in the package. This file also contains tables of values with information about antecedents, rules, and consequents. Library blocks used for the defuzzification method and the antecedents’ connective are extracted directly from the XFL3 specification. However, other library components, as well as the VHDL description style included in the code, depend on the architectural options that the user has selected. Specifically, if the memory type chosen for the antecedents and rules memories is logical block, the description generated for both memories is performed by means of a CASE statement and, in the case of FPGA implementation, CLBs of the programmable device are configured as logic blocks. When the selected option is ROM memory, if the user selects the corresponding option, the synthesis tool can extract the ROM memory to implement these descriptions. Finally, if the selected option is RAM memory, CLBs of the programmable device are configured as distributed RAM memory or embedded RAM memory blocks (BRAMs) can also be used for the implementation of antecedents and rules memories without consuming additional resources of the FPGA. If the system is hierarchical, a VHDL description and a testbench file are generated for each rulebase, which allows checking the control surface obtained for each one of them (as those shown in Fig. 6 of Section V). A structural VHDL description of the hierarchical system is also obtained, with a testbench that allows verifying the corresponding input-output behavior. All the testbench files include the instantiation of the fuzzy system, a process that provides a periodical clock signal, and another process to generate the initial reset signal and a sweep of the input signals used in the simulation of the system.

IV. Hardware Synthesis with XFSG

The second design technique included in *Xfuzzy* is based on XSG. The Xilinx’s tool for the development of DSP systems on FPGAs is integrated into the Matlab environment. It includes a Simulink library (Xilinx Blockset) that provides basic building blocks for digital systems design, as well as the software required to translate Simulink models using these blocks to HDL descriptions that can be implemented on FPGAs. Using this design tool, a new library named *XfuzzyLib* has been generated to accelerate the synthesis of fuzzy systems designed with *Xfuzzy*. This library includes different blocks to implement each of the stages of the active-rule based architecture for fuzzy inference systems described in Section II. Fig. 3a shows the Simulink library browser utility illustrating the fuzzy components grouped by functionalities.

With the help of *XfuzzyLib*, building a fuzzy inference



(a) (b)

Fig. 3. a) Access to *XfuzzyLib* components through the Simulink Library Browser b) Simulink model of a 2-input 1-output FLC that uses product as connective and first-order Takagi-Sugeno as defuzzification method

system requires choosing, interconnecting, and defining the parameters of the needed blocks. Modules in Xilinx Blockset library admit a set of parameters to define their functionality, the size, and the employed arithmetic. Similarly, once the block diagram of a new component in *XfuzzyLib* library has been defined, that element can be encapsulated as a subsystem and a mask can be added to identify its parameters. When the subsystem is instanced in a hierarchical level, parameters can be assigned using numerical values or by means of Matlab variables. Numerical values of these variables can be later defined using the Matlab command window or an ".m" file. System functionality can be verified at any design stage using the facilities from Simulink to generate excitation signals and to capture and display output data.

In addition to the basic building blocks, *XfuzzyLib* also includes elements describing basic fuzzy logic controllers (FLCs) that differ in the number of inputs, the connective used to calculate rule activation degrees, and the defuzzification method. Current version of *XfuzzyLib* incorporates 1-, 2-, and 3-input FLCs using minimum and product as connectives and FuzzyMean, WeightedFuzzyMean, first-order Takagi-Sugeno, and MaxLabel defuzzification methods. When a user needs to develop a fuzzy system tailored to a specific application, these FLCs can be employed or a new architecture can be created by interconnecting basic building blocks. Also it is possible the hierarchical combination of FLCs to define complex fuzzy systems. The block diagram of a 2-input FLC that uses product as connective and first-order Takagi-Sugeno as defuzzification method is shown in Fig. 3b.

Just like basic building blocks from *XfuzzyLib*, blocks describing FLC architectures are fully parametrizable, making it possible to adapt its functionality according to the requirement of a particular application by defining the appropriated parameters. Basically there are two types of parameters: those related to the dimension of the inference system, such as the bus size for inputs, outputs and membership degrees, and other related to the knowledge base of the system, such as the membership functions and the rulebase. In order to facilitate its use to the designer, these parameters correspond to variables and data structures, which can take numeric values using the Simulink graphical interface or an ".m" file. At this design level it is also possible to use the facilities from Matlab environment to verify the functionality of the inference system. Specifically, it results interesting the use of signal sources to explore the universe of discourse of input variables, data acquisition blocks that allow observing the temporal evolution of the system output, and data storage elements that facilitate the graphical representation of control surfaces.

The *xfsg* tool, recently incorporated into *Xfuzzy*, is able to generate the files required to automate this design flow. The Graphical User Interface of this tool is similar to the interface provided by *xfvhdl* shown in Fig. 2. Once all the components have been configured, *xfsg* generates an ".mdl" file containing a Simulink model of the fuzzy system, and an ".m" file with the parameters that define the size and functionality of its components. The generated model includes a "System Generator" block that eases the system implementation by translating the model to different kinds of netlists and generating the bitstream file for the FPGA. XSG is also able to include the appropriated interfaces to co-simulate the hardware implementation of the controller in combination with a mathematical model of the plant under control.

V. Application Example

The above described synthesis tools have been applied to the implementation of a fuzzy system that solves the problem of double integrator, which represents a typical problem in control engineering [40]. The design methodology shown in this section combines the use of specific tools for development of fuzzy systems from the *Xfuzzy* environment, VHDL synthesis tools, and modeling and simulation tools from Matlab and ModelSim. The development of the fuzzy control system with the synthesis tools provided by *Xfuzzy* will be carried out at the different stages illustrated in Fig. 4.

The first stage of the design flow of a fuzzy controller is carried out using the tools included in the *Xfuzzy* design environment [41]. A fuzzy inference module is described in *Xfuzzy* by means of a XFL3 specification, which combines fuzzy rulebases and crisp blocks (to perform inference tasks and to implement arithmetic and logic blocks, respectively). Knowledge rulebases can be directly defined by an expert operator (*xfedit*) or they can be extracted from numerical data using identification algorithms (*xfdm*). For a better performance, the membership functions as well as the rulebases can be simplified (*xfsp*). System parameters can be then adjusted by supervised learning tools (*xfsl*). Functional verification is carried out by two tools included in *Xfuzzy* (*xplot* and *xfsim*). The first one allows analyzing the input-output relation of the system. The second one allows simulating its closed-loop behavior in combination with a Java-codified model of the plant. Fig. 5a shows graphically the description of the controller in *Xfuzzy*. This fuzzy system has been generated by using the identification tool *xfdm* with the help of a set of numerical data. The system uses two rulesbases and a crisp block that performs the arithmetic operation of subtraction. Once validated the XFL3 specification with *xfplot* and *xfsim* (Fig. 6a and 7c, respectively), the synthesis tools presented in this paper are able to generate the files required to start the second design stage.

In this second stage, *xfvhdl* describes the system by a VHDL code that combines different blocks of the VHDL library. The testbench file provided by this synthesis tool allows performing a functional verification of the VHDL description with the ModelSim simulation environment (Fig. 6b). On the other hand, the equivalent system description as a Simulink model provided by *xfsg* is shown in Fig. 5b. The system functionality can be verified at this design phase using the simulation and graphical facilities provided by Simulink and Matlab (Fig. 6c). The similarity between the control surfaces provided by both synthesis tools compared with the graph obtained with the *Xfuzzy* environment (Fig. 6) validates the hardware implementation provided by both design techniques.

As shown in Fig. 4, the design flows for both techniques are different. However, both flows can converge because Simulink allows developing a model where the VHDL code

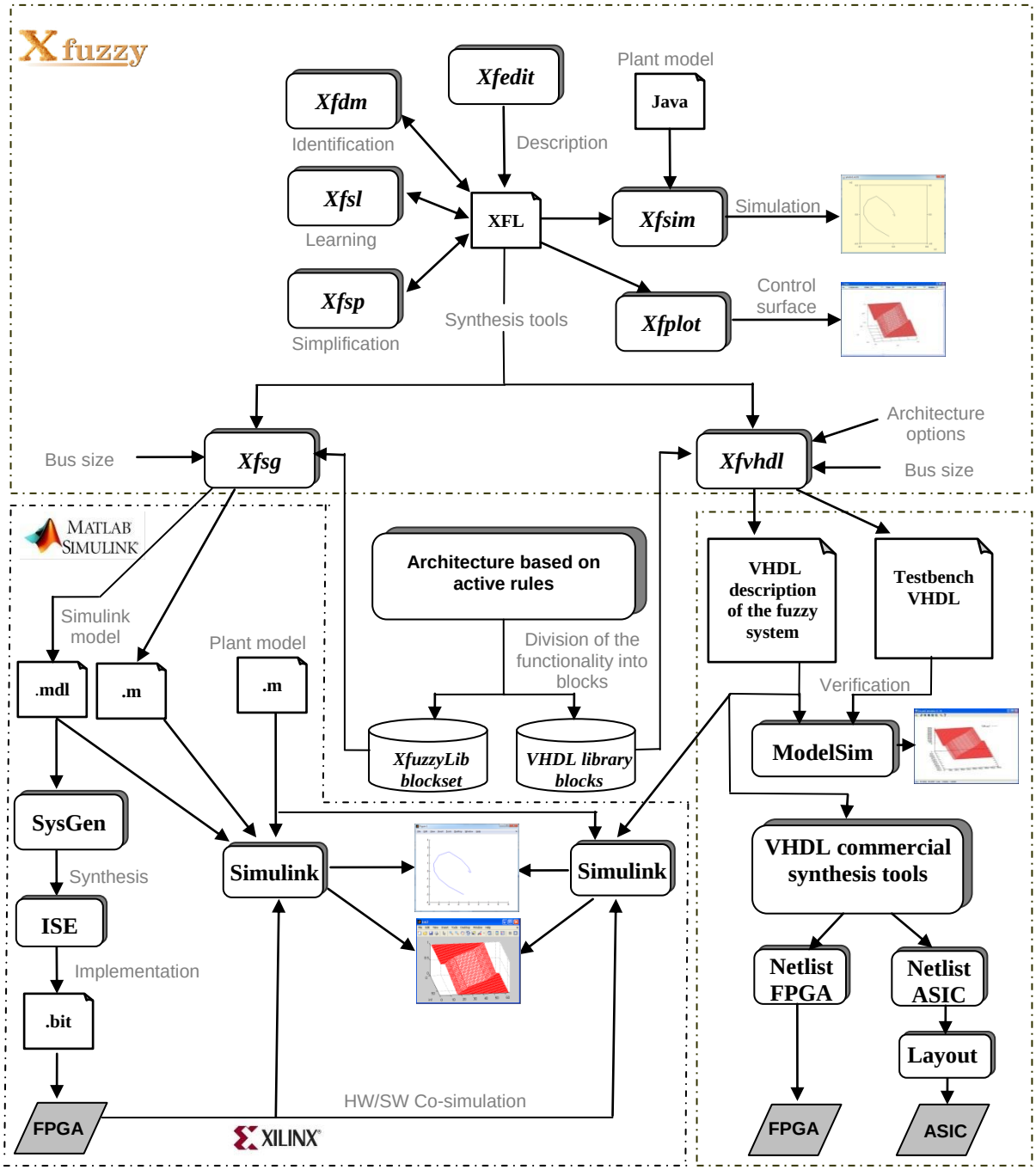
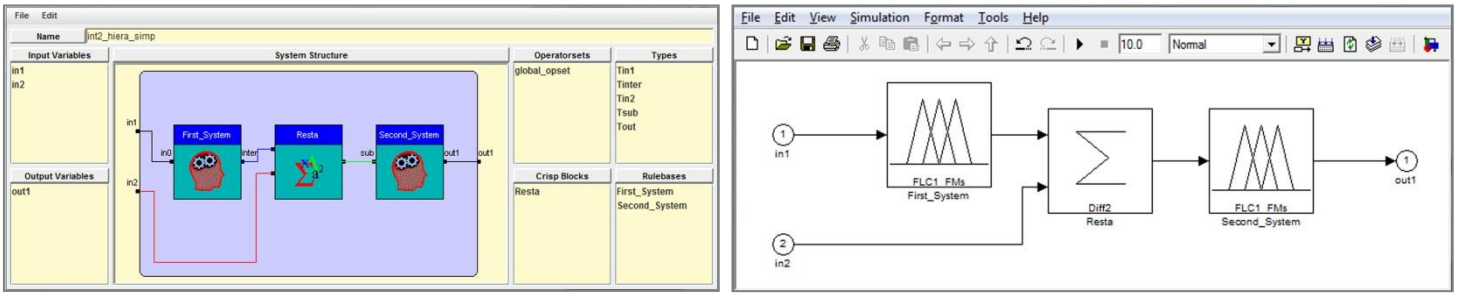


Fig. 4. Fuzzy systems design flow using the hardware synthesis tools provided by *Xfuzzy*

generated by *xfvhdl* can be included in a Black Box block provided by the Xilinx Blockset library, which allows performing an HDL co-simulation where System Generator connects to the ModelSim or ISIM simulators.

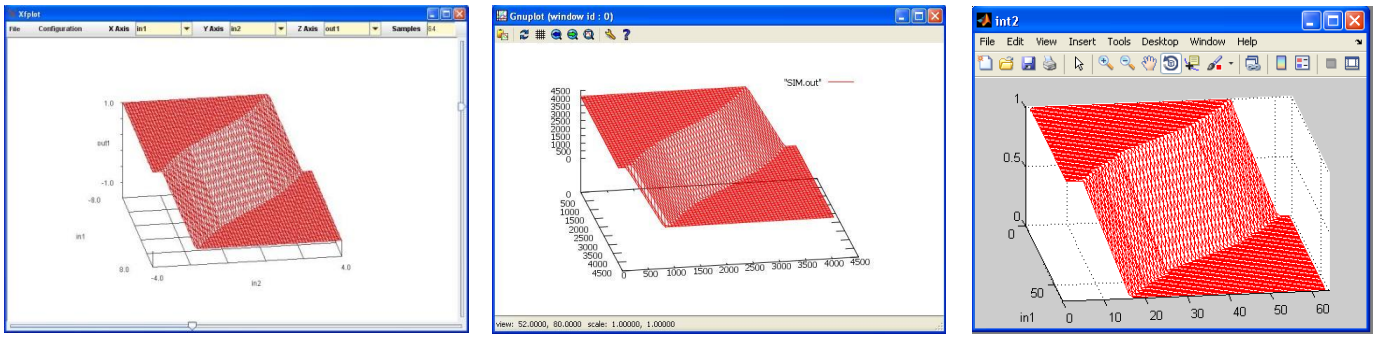
Finally, Simulink also allows carrying out a closed-loop functional verification of any of the implementations described above, combining the co-simulation of a software model of the plant described in Matlab and the hardware controller implementation on an FPGA.

The hardware implementation of this controller using arithmetic techniques for antecedents and ROM memory of distributed type (with twelve bits for input and output precision in all the rules bases) consumes 185 Slices with *xfvhdl* and 259 with *xfsg* (approximately 3% and 4%, respectively, of the Slice resources available in a Spartan 3A FPGA from Xilinx). The controller also employs, for both techniques, 4 of the 20 hardware multipliers available in the FPGA. As it has been described above, different options can be selected for the antecedents and type of memory used in the implementation of the controller. As an example, Tables I and II contain FPGA resource utilization using *xfvhdl* with different implementation options. Table I shows the results after



(a) (b)

Fig. 5. a) *Xfuzzy* graphical representation of the double integrator. b) Simulink model of double integrator generated by *xfsg*



(a) (b) (c)

Fig. 6. Control surfaces generated by: a) *Xfuzzy*. b) ModelSim. c) Matlab/Simulink

implementing the controller with the option of memory storage for antecedents and ROM memory of block type. Table II includes the results obtained using the arithmetic option for antecedents and logic block as type of implementation memory. Both tables detail implementation data relative to the two rulebases and the crisp block of the controller. Implementation results of the VHDL library blocks used in the synthesis of each rulebase are also shown.

Controllers for the double integrator problem implemented with both synthesis tools are able to operate with the 50 MHz clock available at the FPGA development board, which means a control cycle of 120 ns for the controllers considered in this work. Using hardware co-simulation it is possible to evaluate the behavior of the fuzzy controller in a real scenario. As demonstrates the closed-loop simulation shown in Fig. 7, the performance of a 12-bit controller implemented on the FPGA

TABLE I
IMPLEMENTATION RESULTS (SPARTAN 3A, 12 BITS) USING MEMORY FOR ANTECEDENTS AND ROM MEMORY OF BLOCK TYPE

Module	Slices	BRAM	MULT8X18
Double Integrator	8/95	0/14	0/2
+First rulebase	0/40	0/7	0/1
++Control	7/7	0/0	0/0
++Antecedent mem.	2/2	7/7	0/0
++Rule memory	5/5	0/0	0/0
++Rule selector	8/8	0/0	0/0
++Defuzzifier	18/18	0/0	1/1
+Crisp block	14/14	0/0	0/0
+Second rulebase	0/33	0/7	0/1
++Control	7/7	0/0	0/0
++Antecedent mem.	1/1	7/7	0/0
++Rule selector	7/7	0/0	0/0
++Defuzzifier	18/18	0/0	1/1

TABLE II
IMPLEMENTATION RESULTS (SPARTAN 3A, 12 BITS) USING ARITHMETIC OPTION FOR ANTECEDENTS AND LOGIC BLOCK AS TYPE OF MEMORY

Module	Slices	BRAM	MULT8X18
Double Integrator	8/193	0/0	0/4
+First rulebase	0/95	0/0	0/2
++Control	7/7	0/0	0/0
++Arithmetic	47/47	0/0	1/1
++Antecedent mem.	10/10	0/0	0/0
++Rule memory	5/5	0/0	0/0
++Rule selector	8/8	0/0	0/0
++Defuzzifier	18/18	0/0	1/1
+Crisp block	14/14	0/0	0/0
+Second rulebase	0/76	0/0	0/2
++Control	7/7	0/0	0/0
++Arithmetic	41/41	0/0	1/1
++Antecedent mem.	3/3	0/0	0/0
++Rule selector	7/7	0/0	0/0
++Defuzzifier	18/18	0/0	1/1

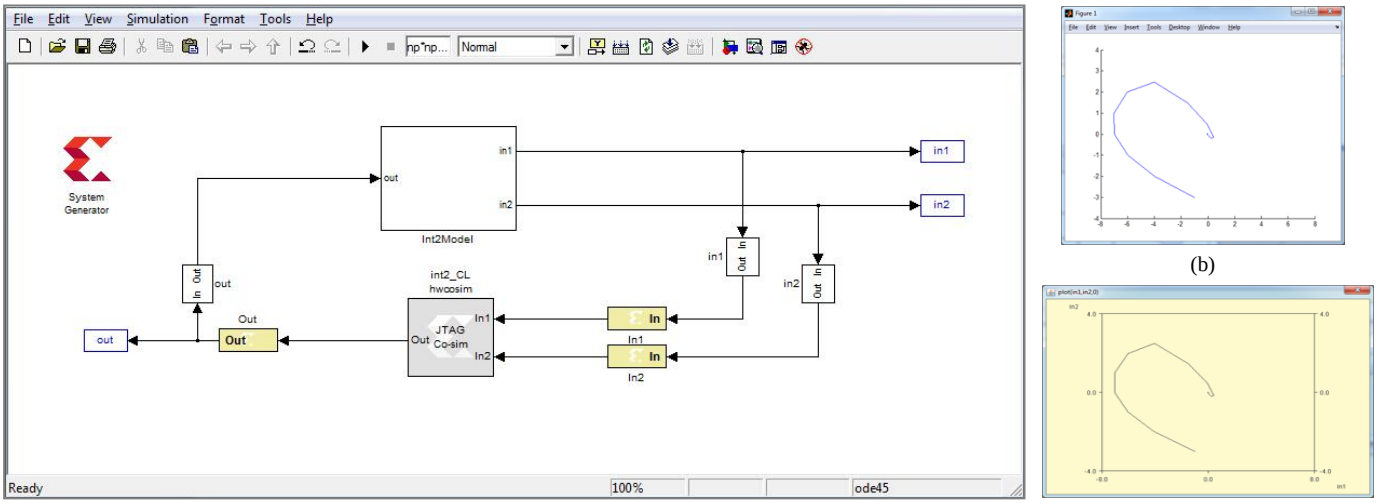


Fig. 7. a) Closed-loop hardware co-simulation. b) Results of the closed-loop verification. c) Results of the closed-loop verification obtained with *xfsim*

board and interacting with a high-level model of the plant (Fig. 7b) is similar to that obtained by the full-precision models used by *xfsim* (Fig. 7c). A quantitative analysis shows a mean error of 0.007 between both results with a standard deviation of 0.004.

VI. Conclusions

Two design strategies for the automatic synthesis of fuzzy inference systems have been presented in this paper. They demonstrate that the availability of a design flow, supported by the use of parameterized cell libraries and CAD tools, considerably speeds up the hardware implementation of fuzzy systems, facilitating the exploration of the design space for a given application. One of the described tools is focused to hardware implementations of fuzzy systems on Xilinx's FPGAs, while the other one provides synthesizable VHDL code for ASICs and FPGAs. Compared to previous releases of hardware synthesis tools included in *Xfuzzy* [41] [42], the tools described in this paper provide an improved functionality of most of the components included in the VHDL and Simulink libraries, such as the generation of families of membership functions of type "sh_triangular" (where the first and last functions are Z- and S-shaped, respectively), as well as new operators that implement arithmetic and logical crisp functions, and a new defuzzification block for first-order Takagi-Sugeno's systems. Both design libraries have been also revised in order to increase their operational speed and reduce the resource consumption. Additionally, improved graphical interfaces that consider the new features of the tools have been completely integrated into the *Xfuzzy* environment. Finally, the most important advantage of the new release is the possibility of direct implementation of hierarchical fuzzy systems. As demonstrated by their application to an FPGA design example, the choice between the two design strategies proposed in this paper allows obtaining an adequate tradeoff between "high system performance" (usually reached by the VHDL based option) and "short design time" (provided by the XSG approach).

Acknowledgment

This work was partially funded by Spanish Ministerio de Economía y Competitividad under the Project TEC2011-24319 and by Junta de Andalucía under the Project P08-TIC-03674 (both with support from FEDER), and by the European Community through the MOBY-DIC Project FP7-INFSO-ICT-248858 (www.mobydic-project.eu). P. Brox is supported under the post-doctoral program "Juan de la Cierva" from the Spanish Government.

References

[1] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 1, January 1973, pp. 28-44.

[2] T. J. Ross, *Fuzzy Logic with Engineering Applications*, 2nd ed. Hoboken, NJ: Wiley, 2004.

[3] J. Jarris, *Fuzzy logic applications in engineering science*, Springer Verlag, 2006.

[4] R.-E. Precup and H. Hellendoorn, "A survey on industrial applications of fuzzy control", *Computers in Industry*, vol. 62, no. 3, April 2011, pp. 213-226.

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

[6] K. Basterretxea and I. del Campo, *Electronic hardware for fuzzy computation*, in A. Laurent and M-J. Lessot, editors, Scalable Fuzzy Algorithms for Data Management and Analysis: Methods and Design, Information Science Reference, 2009.

[7] A. H. Zavala and O. C. Nieto, "Fuzzy Hardware: A Retrospective and Analysis", *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 4, August 2012, pp.623-635.

[8] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, Design Tools, and Application Domains of FPGAs", *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, August 2007, pp. 1810-1823.

[9] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in Industrial Control Applications", *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, May 2011, pp. 224-243.

[10] N. Sulaiman, Z.A. Obaid, M.H. Marhaban, and M. N. Hamidon, "Design and Implementation of FPGA –Based Systems – A Review", *Australian Journal of Basic and Applied Sciences*, vol. 3, no. 4, 2009, pp. 3575-3596.

[11] M. McKenna and B.M. Wilamowski, "Implementing a fuzzy system on a field programmable gate array", in *Proc. Int. Joint Conf. on Neural Networks*, July 2001, pp. 189-194.

[12] D. N. Oliveira, A. P. de Souza Braga, and O. da Mota Almeida, "Fuzzy Logic Controller Implementation on an FPGA using VHDL", in *Proc. Fuzzy Information Processing Society (NAFIPS)*, 2010 Annual Meeting of the North American, July 2010, pp. 1-6.

[13] G. Sakthivel, T. S. Anandhi, and S. P. Natarajan, "Real Time Implementation of a Fuzzy Logic Controller on FPGA Using VHDL for DC Motor Speed Control", *International Journal of Engineering Science and Technology*, vol. 2, no. 9, 2010, pp. 4511-4519.

[14] System Generator for DSP User Guide, v10.1, Xilinx Inc., 2008. Available: http://www.xilinx.com.

[15] M. Bahoura and H. Ezzaidi, "FPGA-implementation of a sequential adaptive noise canceller using Xilinx System Generator", in *Proc. Int. Conf. on Microelectronics*, December 2009, pp. 213-216.

[16] A. Toledo, P. Navarro, F. Soto, J. Suardíaz, and C. Fernández, "Experiences on developing computer vision hardware algorithms using Xilinx system generator", *Microprocessors and Microsystems, Special Issue on FPGAs: Case Studies in Computer Vision and Image Processing*, vol. 29, issues 8-9, November 2005, pp. 411-419.

[17] R. Sepúlveda, O. Montiel, G. Lizágarra, and O. Castillo, "Modeling and Simulation of the Defuzzification Stage of a Type-2 Fuzzy Controller using the Xilinx System Generator and Simulink", *Evolutionary Design of Intelligent Systems*, vol. 257, Springer-Verlag, 2009, pp. 309-325.

[18] Y. Kung, C. Huang, and M. Tsai, "FPGA Realization of an Adaptive Fuzzy Controller for PMLSM Drive", *IEEE Transactions on Industrial Electronics*, vol.56, no.8, August 2009, pp. 2923-2932.

[19] F. Taeed, Z. Salam, and S. M. Ayob, "FPGA Implementation of a Single-Input Fuzzy Logic Controller for Boost Converter with the Absence of an External Analog-to-Digital Converter", *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, February 2012, pp. 1208-1217.

[20] C. Huang, W. Wang, and C. Chiu, "Design and Implementation of Fuzzy Control on a Two-Wheel Inverted Pendulum", *IEEE Transactions on Industrial Electronics*, vol.58, no.7, July 2011, pp. 2988-3001.

[21] H. Huang and C. Tsai, "FPGA Implementation of an Embedded Robust Adaptive Controller for Autonomous Omnidirectional Mobile Platform", *IEEE Transactions on Industrial Electronics*, vol.56, no.5, May 2009, pp. 1604-1616.

[22] S. Sánchez-Solano, A. Cabrera, I. Baturone, F.J. Moreno-Velo, and M. Brox, "FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications", *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, August 2007, pp.1937-1945.

[23] Y. Fu, H. Li, and M. E. Kaye, "Hardware/Software Codesign for a Fuzzy Autonomous Road-Following System", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 6, November 2010, pp. 690-696.

[24] C.-F. Juang and J.-S. Chen, "Water bath temperature control by a recurrent fuzzy controller and its FPGA implementation", *IEEE Transactions on Industrial Electronics*, vol. 53, no. 3, June 2006, pp. 941-949.

[25] G. Louverdis and I. Andreadis, "Design and Implementation of a Fuzzy Hardware Structure for Morphological Color Image Processing", *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no. 3, March 2003, pp.277-288.

[26] C. Kavka, M. Crespo, W. Geng, and F. Zhong, "A Fuzzy Controller Development Tool based on Evolutionary Techniques", in *Proc. of the 1999 Congress on Evolutionary Computation*, July 1999, pp. 2145-2150.

[27] J.M. Alonso, L. Magdalena, and S. Guillaume, "KBCT: a knowledge extraction and representation tool for fuzzy logic based systems", in *Proc. IEEE Int. Conf. on Fuzzy Systems*, vol.2, July 2004, pp. 989-994.

[28] E. Moreira and A. Sousa, "FEUP Fuzzy Tool II Improved tool for education and embedded control", in *Proc. CISTI'2010 - 5ª Conferencia Ibérica de Sistemas y Tecnologías de Información*, June 2010, pp. 1-6.

[29] S. Guillaumea and B. Charnomordicb, "Learning interpretable fuzzy inference systems with FisPro", *Information Sciences, Special Issue on Interpretable Fuzzy Systems*, vol. 181, no. 20, October 2011, pp. 4409-4427.

[30] T. Hollstein, S. K. Halgamuge, and M. Glesner, "Computer-Aided Design of Fuzzy Systems Based on Generic VHDL Specifications", *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 4, November 1996, pp. 403-417.

[31] R. G. Carvajal, A. Torralba, and L. G. Franquelo, "AFAN: a tool for the automatic synthesis of neural and fuzzy controllers with architecture optimization", in *Proc. International Symposium on Circuits and Systems*, June 1997, vol. 1, pp. 637-640.

[32] M. Re, M. Salmeri, and G. Cardarilli, "A CAD environment for fuzzy systems hw/sw mapping", in *Proc. International Symposium on Circuits and Systems*, May 2000, pp. 221-224.

[33] D. Kim and In-Hyun Cho, "FADIS: An Integrated Development Environment for Automatic Design and Implementation of FLC", in *Proc. 1997 Annual Meeting of the North American Fuzzy Information Processing Society*, September 1997, pp. 33-39.

[34] A. Bakhti and L. Benbaouche, "Simulink-DSP Co-Design of a Fuzzy Logic Controller", *Industrial Electronics Society Annual Conference*, vol.1, November 2006, pp. 4587-4592.

[35] I. H. Altas and A.M. Sharaf, "A Generalized Direct Approach for Designing Fuzzy Logic Controllers in Matlab/Simulink GUI Environment", *International Journal of Information Technology and Intelligent Computing, Int. J. IT&IC*, no.4 vol.1, 2007.

[36] M. Shahrieel, S. Najib, E. Chee, I. Azmira, and Mohd Hendra, "Comparison of Fuzzy Control Rules using MATLAB Toolbox and Simulink for DC Induction Motor-Speed Control", in *Proc. 2009 International Conference of Soft Computing and Pattern Recognition*, December 2009, pp. 711-715.

[37] ChanghuaLu and J. Zhang, "Design and Simulation of a Fuzzy-PID Composite Parameters' Controller with MATLAB", in *Proc. 2010 International Conference on Computer Design and Applications* (ICCDA 2010), June 2010, pp. 308-311.

[38] O. Kobyrynka, Y. Stekh, and O. Markelov, "Comparison analysis of methods implemented in MATHLAB for fuzzy logic algorithms", in *Proc. 2011 CAD Systems and Microelectronics*, February 2011, pp. 239-240.

[39] Xfuzzy: Fuzzy Logic Design Tools, IMSE-CNM, CSIC. Available: http://www.imse-cnm.csic.es/Xfuzzy

[40] I. Baturone, M. C. Martínez-Rodríguez, P. Brox, A. Gersnoviez, and S. Sánchez-Solano, "Digital Implementation of Hierarchical Piecewise-Affine Controllers", in *Proc. 20th International Symposium on Industrial Electronics (ISIE 2011)*, June 2011, pp.1497-1502.

[41] I. Baturone, F. J. Moreno-Velo, S. Sánchez-Solano, A. Barriga, P. Brox, A. Gersnoviez, and M. Brox, "Using Xfuzzy Environment for the Whole Design of Fuzzy Systems", *in Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, July 2007, pp. 517-522.

[42] S. Sánchez-Solano, M. Brox, E. del Toro, P. Brox, and I. Baturone, "Model-Based Design Methodology for Rapid Development of Fuzzy Controllers on FPGAs", *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, 2012, p. 1.

María Brox received the B.Sc. degree (with honors) in physics from the University of Córdoba, Córdoba, Spain, in 2004, and the M.Sc. degree in microelectronics from the University of Seville, Seville, Spain, in 2008.

From 2005 to 2007, she held a postgraduate fellowship from the Spanish Government with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain. She is currently an Assistant Professor with the Department of Computer Architecture, University of Córdoba, Córdoba, Spain. Her research area is the development of automatic CAD tools for the design of embedded fuzzy controllers on FPGAs.

Santiago Sánchez-Solano received the B.Sc. (with honors) and Ph.D. degrees from the University of Seville, Seville, Spain, in 1980 and 1990, respectively, both in physics.

After six years as a System Analyst with the Computer Center, University of Seville, Spain, he joined the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, where he is currently a Scientific Researcher. He is the coauthor of two books and 150 scientific papers and has participated in 25 research projects funded by different organisms, acting in seven of them as lead researcher. His research interests include very large scale integration design, computer-aided-design tools for microelectronic design, and hardware implementation of neuro-fuzzy systems.

Ernesto del Toro received the B.Sc. degree in automation engineering and M.Sc. degree in electronics from the Instituto Superior Politécnico J.A.E. of Havana (CUJAE), Havana, Cuba, in 2004 and 2007, respectively.

He held a MAEC-AECID PhD scholarship from the Spanish Government in the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain, from 2008 to 2011. Currently, he is a Professor of electronics and a Research Assistant with the Microelectronics Research Center (CIME-CUJAE), Havana, Cuba. His research interests include embedded computing, hardware/software codesign and algorithm acceleration.

Piedad Brox received the B.Sc. degree from the University of Córdoba, Córdoba, Spain, in 2002, and the Ph.D. degree (with honors) from the University of Seville, Seville, Spain, in 2009, both in physics.

Since 2002, she has been with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain, or with the University of Seville. Currently, she is a Postdoctoral Researcher under the "Juan de la Cierva" program funded by the Spanish Government. Her research areas include the design and implementation of neuro-fuzzy systems and its application in image processing, and digital implementation of embedded controllers.

Francisco J. Moreno-Velo received the B.Sc. degree in physics and the B.Sc. and Ph.D. degrees in computer science from the University of Seville, Seville, Spain, in 1995, 1996 and 2003, respectively.

From 1996 to 1999, he was an Assistant Professor with the Department of Applied Physics and Electrical Engineering, University of Huelva, Huelva, Spain. From 2000 to 2003, he was a Postgraduated Research Fellow at the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville. Currently, he is an Associate Professor with the Department of Information Technologies, University of Huelva. His current research interests include fuzzy systems, softcomputing techniques, development of computer-aided design tools for fuzzy systems, and compiler design.