FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications

Santiago Sánchez-Solano, Alejandro J. Cabrera, Iluminada Baturone, Francisco J. Moreno-Velo, and María Brox

***Abstract*—Fuzzy-logic-based inference techniques provide efficient solutions for control problems in classical and emerging applications. However, the lack of specific design tools and systematic approaches for hardware implementation of complex fuzzy controllers limits the applicability of these techniques in modern microelectronics products. This paper discusses a design strategy that eases the implementation of embedded fuzzy controllers as systems on programmable chips. The development of the controllers is carried out by means of a reconfigurable platform based on field-programmable gate arrays. This platform combines specific hardware to implement fuzzy inference modules with a general-purpose processor, thus allowing the realization of hybrid hardware/software solutions. As happens to the components of the processing system, the specific fuzzy elements are conceived as configurable intellectual property modules in order to accelerate the controller design cycle. The design methodology and tool chain presented in this paper have been applied to the realization of a control system for solving the navigation tasks of an autonomous vehicle.**

***Index Terms*—Autonomous vehicles, embedded systems, field-programmable gate arrays (FPGAs), fuzzy control, intellectual property (IP).**

I. Introduction

FUZZY logic provides a mathematical framework to deal with the uncertainty and the imprecision typical of the human reasoning system. One of its main characteristics is the capability to describe the behavior of a complex system in a linguistic way by means of IF–THEN rules similar to those employed in natural language [1]. The facility of fuzzy logic controllers (FLCs) to capture the knowledge of human experts and translate it into robust control strategies without the need of a mathematical model of the system under control has led to a significant increase in the number of control applications using fuzzy inference techniques in the last 25 years [2]. This has motivated the development of different approaches to implement fuzzy control systems. These approaches range from fully software or hardware solutions to hybrid strategies that allow reaching adequate tradeoffs between flexibility and inference speed [3]. Hybrid realizations require a processor for software task execution and dedicated hardware for implementing complex time-consuming tasks, i.e., usually the fuzzy inference process.

Many of the emerging applications of fuzzy controllers in fields such as autonomous mobile robots, artificial vision systems, or traffic control in the Internet demand area, power, speed, or cost requirements that can only be satisfied by implementing them as embedded systems. The continuous advances in microelectronic industry and its integration capabilities allow the realization of such complex circuits on a single silicon chip [4]. However, the rapid evolution of silicon technologies has not only increased the density of application-specific integrated circuits (ASICs) but also allowed that programmable logic devices like field-programmable gate arrays (FPGAs) reduce their die sizes and therefore their cost, so hardware platforms based on FPGAs can be used not only as rapid prototyping approaches but also as final solutions that greatly shorten the time-to-market of new consumer products. Modern FPGA devices include complex specific resources such as memory blocks, hardware multipliers, clock management circuits, and high-speed interface circuits. Most of them also include soft cores for processors, input–output (I/O) peripherals, memory controllers, etc., which facilitate the development of general-purpose processing systems. Some examples are the MicroBlaze system from Xilinx and the Nios processor from Altera. As a consequence, the availability of low-cost large-capacity FPGAs, many standard system components described as intellectual property (IP) modules, and powerful computer-aided design (CAD) tools make possible nowadays the whole development of a system on a programmable chip (SoPC) [5]. Depending on the application and the number of chips to be fabricated, this approach may become more attractive than ASIC solutions.

A novel strategy for the implementation of embedded fuzzy controllers as SoPC is presented in this paper. Section II briefly explains the fundamentals of fuzzy control and the different implementation alternatives reported in the literature. The main components of a reconfigurable platform for the development of hybrid hardware/software (HW/SW) fuzzy controllers are described in Section III. The design flow proposed, which combines standard FPGA implementation tools and a specific environment for the development of fuzzy controllers as IP modules, is outlined in Section IV. As an example of using this design strategy, Section V shows the development of a

Manuscript received June 25, 2006; revised March 29, 2007. This work was supported in part by the Spanish Ministry of Education and Science under Projects TEC2005-04359/MIC and DPI2005-02293, and in part by the Andalusian Regional Government under Projects TIC2006-635 and TEP2006-375.

S. Sánchez-Solano and I. Baturone are with the Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, CNM-CSIC, 41012 Seville, Spain (e-mail: santiago@imse.cnm.es; lumi@imse.cnm.es).

A. J. Cabrera is with the Instituto Superior Politécnico José Antonio Echeverría, CUJAE, Havana 10400, Cuba (e-mail: alex@electrica.cujae.edu.cu).

F. J. Moreno-Velo is with the Universidad de Huelva, 21071 Huelva, Spain (e-mail: francisco.moreno@diesia.uhu.es).

M. Brox is with the University of Córdoba, 14071 Córdoba, Spain (e-mail: mbrox@uco.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2007.898292

control system for parking an autonomous mobile robot, and Section VI discusses the obtained experimental results. Finally, main conclusions are summarized in Section VII.

II. HW/SW Implementation of Fuzzy Controllers

Fuzzy controllers employ the same input and output variables as their conventional counterparts. The difference between both approaches is that, in most of the practically used conventional controllers, the output is obtained as a linear combination of inputs, while in the fuzzy approach, the control heuristic is defined by a set of rules that employ linguistic variables represented by fuzzy sets. As a consequence, the (usually nonlinear) control surface provided by a fuzzy controller can be locally modified by changing the rules that affect the corresponding areas in the universes of discourse. In addition, the use of fuzzy logic allows the development of robust controllers able to support great perturbations and to provide soft output variations with a small number of rules.

The mathematical framework of fuzzy inference techniques provides systematic and deterministic algorithms that can be easily described with high-level programming languages or implemented as electronic circuits. Thus, there are basically two alternatives for fuzzy controller implementation: one of them is based on software and the other on hardware. Software solutions offer flexibility as one of their main features (the designer can choose any type of fuzzy sets, operators, and rulebases). For this reason, many of the first fuzzy controllers were implemented in software on general-purpose computers by describing the fuzzy algorithm with a high-level programming language. In applications in which size, weight, power, or cost are constrained, as occurs with consumer electronics, standard microcontrollers have been usually employed. Many software tools that ease the development of FLCs by generating optimized code for different families of microcontrollers were developed in the last decade of the past century. MicroFPL, TIL Shell, and Fuzzy CLIPS from Togai InfraLogic; FIDE from Aptronix; and FuzzyTECH from Inform are some of them.

The main drawback of software implementations of FLCs is speed limitation due to the sequential program execution and the fact that standard processors do not directly support many fuzzy operations, such as minimum or maximum. In order to reduce the lack of fuzzy operations, some developments modifying the architecture of standard processors to support fuzzy computation have been carried out [6], [7]. The 68HC12 microcontroller from Motorola [8] and the ST Five microcontroller family from ST Microelectronics [9] are commercially available devices that use these techniques. Although the specific hardware added to these devices speed up fuzzy computation by at least one order of magnitude over standard processors, these software solutions are still not fast enough for some real-time applications, where a dedicated hardware structure must be used [10].

Since the initial fuzzy hardware proposals from Togai and Watanabe [11] and Yamakawa and Miki [12] in the mid 1980s, many microelectronic implementations of fuzzy controllers have been described in the last years [13]. Both digital and analog design techniques have been employed, with the digital approaches being the most widely used due to the availability of well-established design methodologies and supporting CAD tools. In order to accelerate fuzzy computation, many hardware solutions implement the inference module or some of its units on a dedicated integrated circuit, so they need to be connected to a general-purpose processing system to complete their operation. These hardware implementations, which are known as fuzzy accelerators or fuzzy coprocessors, were very popular at the end of the 1990s. Some examples of commercially available fuzzy coprocessors are the SAE 81C99 and SAE 81C991 families from Siemens [14]; the FP1000, FP3000, and FP5000 fuzzy coprocessors from Omron [15]; and the W.A.R.P. architecture from ST Microelectronics [16]. Nowadays, fuzzy coprocessors are seldom used, and most of them have disappeared from the market as a consequence of the increasing speed of conventional processors, which makes it possible to execute the simple fuzzy algorithms required by many applications.

Fully hardware realizations of FLCs implemented on FPGAs [17]–[19] or ASICs [20], [21] achieve a very high inference speed but are characterized by their lack of flexibility because some limitations must be adopted in order to obtain a cost-effective dedicated hardware. Typically, these realizations implement a specific fuzzy inference algorithm for a specific application, so no configuration options are possible. The high performance of these FLCs is usually achieved at the expense of a long design time.

Halfway between software and hardware implementations, the use of hybrid HW/SW techniques permits appropriate tradeoffs between the advantages and drawbacks of both approaches. The development of a high-performance HW/SW fuzzy controller requires an efficient partition of the different tasks to execute in order to obtain a flexible and high-speed system. According to this idea, complex time-consuming tasks [usually the whole fuzzy inference module (FIM)] must be implemented in hardware, while those tasks related to system configuration and nonfuzzy computations are better executed by software [22].

Recent advances in silicon technologies allow integrating a complex fuzzy-logic-based control system on a single chip. On the other hand, general and specific resources available on current FPGAs make possible its implementation as SoPC. The hybrid HW/SW strategy for the implementation of embedded fuzzy controllers as SoPC employed in this paper is illustrated in Fig. 1. It is based on the use of a general-purpose processor available as an IP module and the employment of application-specific FIMs (also developed as IPs) to accelerate the inference tasks [23]. The whole controller can be easily implemented on an FPGA by combining standard FPGA synthesis and implementation tools and a specific CAD environment for the development of fuzzy systems, thus considerably reducing the design cycle of new controllers.

III. Development Platform for Hybrid Fuzzy Systems

In order to implement the above-introduced strategy, a development platform for fuzzy controllers has been defined. This

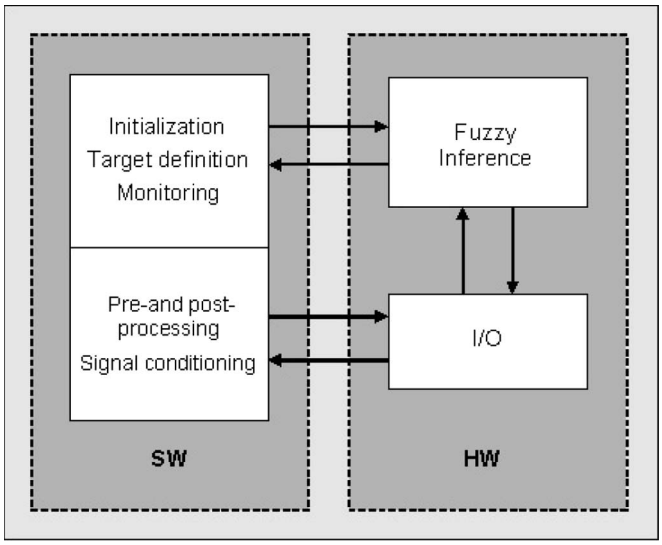

Fig. 1. Hardware–software partitioning of a fuzzy-logic-based control system.

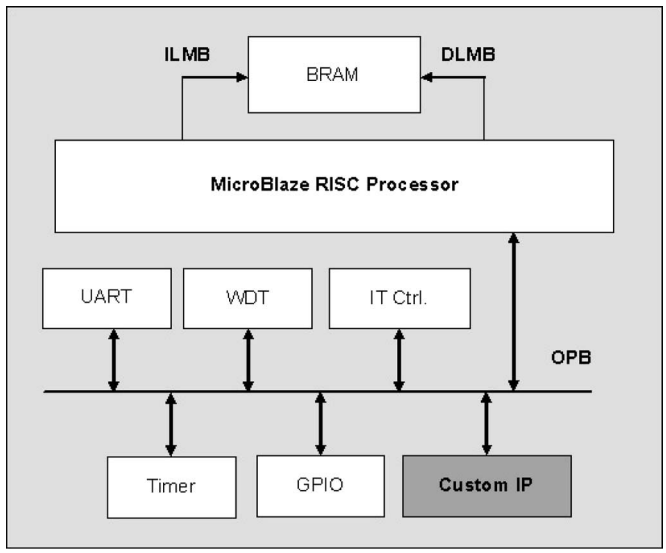

Fig. 2. MicroBlaze-based processing system.

platform integrates all the physical and logical elements as well as the procedures required to build embedded fuzzy controllers on programmable logic devices. The physical components including the configurable IP modules of the processing system, the specific architecture for FIMs, and the FPGA development board required to support the SoPC will be described in this section. The FPGA design tools and the fuzzy system development environment making up the logical components of the platform will be discussed in the next section jointly with the proposed design flow.

A. MicroBlaze Processing System

According to the aforementioned idea, the processing system will perform all the controller tasks that do not need specific hardware resources. The selected element has been the MicroBlaze IP module from Xilinx. MicroBlaze is a 32-bit Reduced Instruction Set Computer processor based on Harvard architecture that can be implemented into Xilinx Spartan II, Spartan 3, and Virtex FPGA families [24]. A simplified block diagram of the MicroBlaze system is shown in Fig. 2. Data and instruction buses are further divided into local buses (to rapid connection of dual-port block memory structures available in the FPGAs) and peripheral buses based on the IBM's standard on-chip peripheral bus (OPB). There are many IP modules of configurable peripherals compatible with this standard (input/output ports, timers, Universal Asynchronous Receiver Transmitter (UART), interrupt controllers, etc.) that allow configuring the system according to application requirements. Design tools included in Xilinx's Embedded Development Kit (EDK) provide software drivers that ease the use of MicroBlaze peripherals as well as facilities for converting custom hardware into OPB-compatible IP modules (as shown in the shaded box of Fig. 2).

B. FIM Architecture

The FIMs required by hybrid solutions will be implemented on hardware by using the resources available in FPGA. These

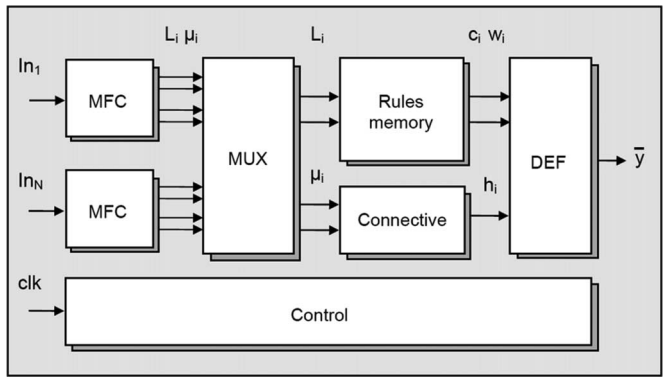

Fig. 3. Block diagram of FIMs.

modules follow the architecture and design methodology previously developed in [25] for fuzzy ASICs. Some of the keys that allow this architecture to provide low-cost and high-speed digital FIMs are the following: 1) to evaluate only the contribution of the active rules; 2) to restrict the shapes of the fuzzy set membership functions; and 3) to use simplified defuzzification methods. Active rule processing contributes to increase the speed of the fuzzy inference process by reducing the number of rules to be evaluated to those whose contribution is a nonzero value. Constraining to two the overlapping degree of input fuzzy set reduces to 2^I the number of active rules to be processed instead of the total number of rules L^I of a system with I inputs and L fuzzy sets per input. In addition, if the input fuzzy sets are also constrained to be normalized piecewise linear functions, the membership function degree can be obtained by simple arithmetic methods. The use of simplified defuzzification methods (instead of conventional ones) also contributes to increase system speed and to reduce hardware resources consumption.

Fig. 3 shows a block diagram of the FIM architecture used in the development platform. In each operation cycle, the membership function circuits (MFCs) evaluate the input values and provide pairs of "label, activation-level" (L_i, μ_i) values. 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 (MUX). In each clock cycle, the membership degrees μ_i from rule antecedents are combined within the connective-antecedent operator circuit to calculate the activation level h_i of the rule, while the antecedent labels L_i address the rule memory location containing the parameters (c_i, w_i) that define its corresponding consequent. A configurable defuzzifier block (DEF) is used to obtain the crisp output value $\overline{y}$, whereas a control block regulates the timing of the fuzzy inference process.

One advantage of this architecture arises from the availability of different circuit realizations for the blocks in Fig. 3, which provides a high configuration degree to the fuzzy modules used in the controller. MFCs can be implemented using either memory- or arithmetic-based approaches. Memory-based MFCs store the labels and the membership function degrees corresponding to each point of the input universe of discourse. The main advantages of this approach are its high operation speed and the possibility to use unrestricted fuzzy set shapes. Its major drawback is the exponential increase of memory consumption when the number or the resolution of inputs grows up. Arithmetic-based MFCs employ an arithmetic circuit to implement normalized triangular functions using the breakpoints and slopes stored in a common parameter memory, thus reducing the memory requirements. Implementation of fuzzy operators and defuzzification methods can also be selected by the designer among several options. The connective block in Fig. 3 that represents the conjunctive nexus between rule antecedents ("and" clause) can be implemented by a minimum or product operator. Finally, depending on the kind (interpolating or decision making) of fuzzy module, different defuzzifier blocks can be employed. Fig. 4(a) shows the block diagram of a circuit that implements the equation

$$\overline{y} = \frac{\sum_i h_i \cdot c_i \cdot w_i}{\sum_i h_i \cdot w_i} \tag{1}$$

where h_i's are the activation degrees of the rules, c_i's are crisp values representing the rule conclusions, and w_i's are their corresponding weight factors. According to the meaning of the w_i parameter, some well-known defuzzification methods like Fuzzy Mean, Weighted Fuzzy Mean, or Center of Sums can be implemented by this circuit. As a particular case, only one multiplier is needed for Fuzzy Mean implementation $(w_i = 1)$. The architecture also has a defuzzification option for decision-making systems. Fig. 4(b) shows the schematic of a defuzzifier named "MaxLabel" in which the rule consequent c_i with the highest activation degree h_i will be considered as the crisp system output.

Regarding timing considerations, memory-based MFCs obtain antecedent activation levels in only one clock cycle, whereas arithmetic MFCs need as many clock cycles to perform its operation as membership functions are defined. One clock cycle is required by the active-rule selection mechanism to evaluate each active rule (the number of active rules depends on the number of inputs and the overlapping degree of input membership functions). Finally, using a sequential division circuit with shift subtract/add nonrestoring technique, the division

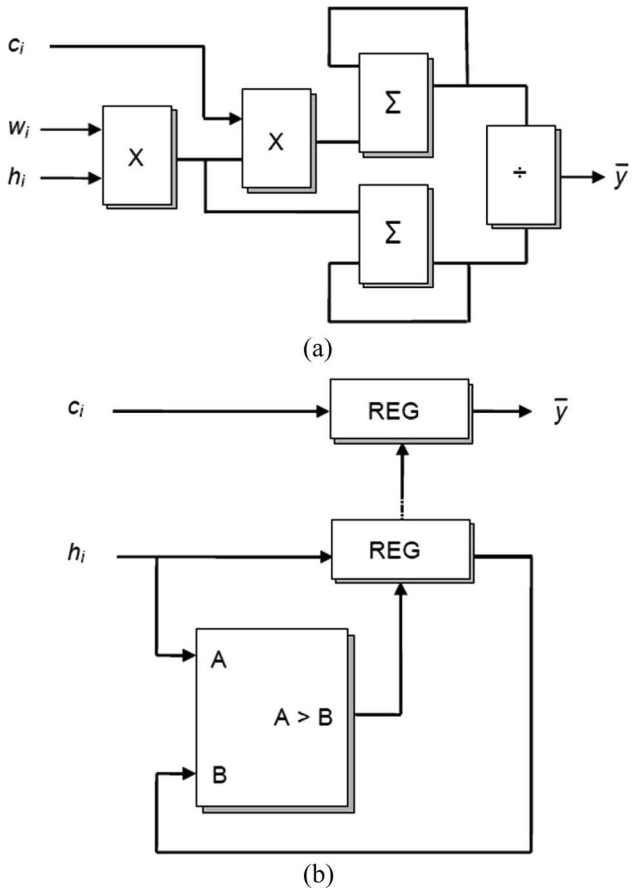

Fig. 4. (a) Block diagram of a configurable defuzzifier module. (b) Defuzzifier for decision-making systems.

at the defuzzifier stage needs a clock cycle for each bit of the quotient (this division can be eliminated when using normalized membership functions and the product operator as connective). Introducing pipeline stages among the three main blocks of the architecture, the number of required clock cycles to produce a control output is limited by the maximum between the number of active rules and the number of cycles required to perform the operations of the fuzzifier and defuzzifier interfaces.

C. FPGA Development Board

In order to give physical support to the development platform, a Spartan-3 Starter Board (from Digilent Inc.) has been employed. This board includes a 1000K gate Spartan-3 with 24 18-bit multipliers and 432 Kb of block RAM. It also contains onboard I/O devices (slide switches, pushbuttons, light-emitting diodes, and seven-segment display), serial and JTAG ports, and 1-MB asynchronous SRAM.

IV. CAD Tools and Design Flow

A second reason to select the above FIM architecture in the proposed platform is the availability of CAD tools that make the design and implementation of FIMs easy. These tools are included in the *Xfuzzy* environment [26]. As described in this section, the output of the synthesis facilities provided by *Xfuzzy* can be combined with standard FPGA tools in order to obtain

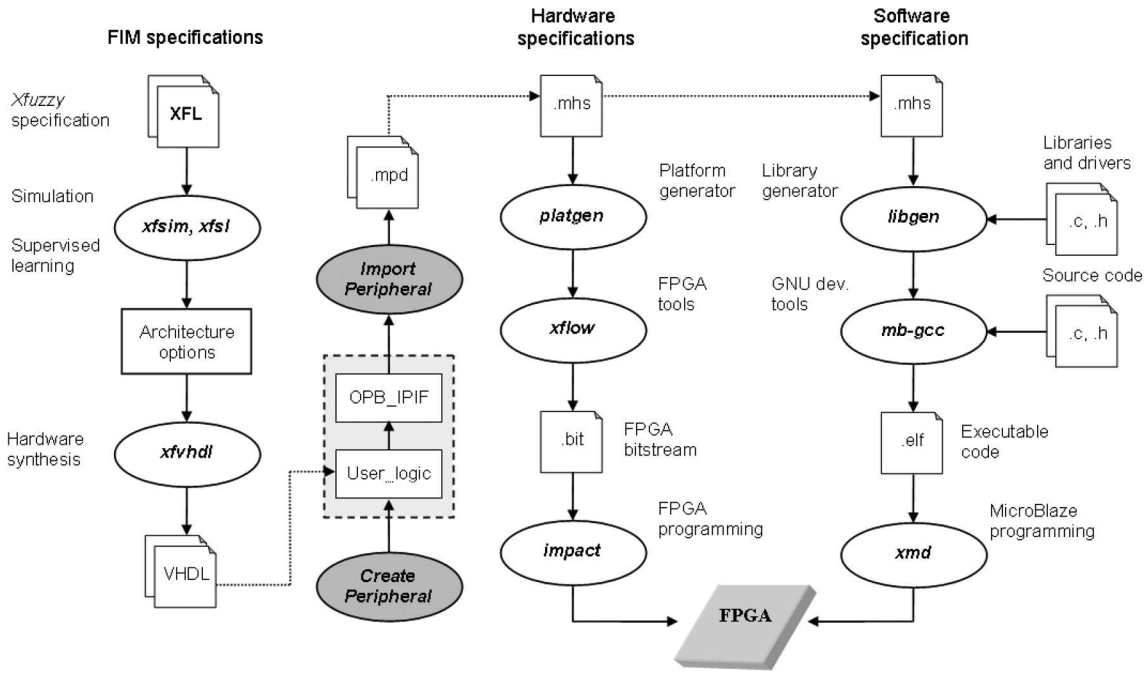

Fig. 5. Design flow of FLCs combining Xilinx and *Xfuzzy* tools.

an IP module that can be used in an FPGA-based embedded fuzzy controller.

A. Xfuzzy Environment

The *Xfuzzy* design environment facilitates the different development stages of fuzzy systems by providing description, simulation, learning, and simplification tools that permit the fuzzy system to be defined, verified, and optimized. The environment also includes synthesis tools that provide software or hardware implementations of fuzzy inference systems [27]. The common specification language of Xfuzzy, i.e., XFL, used by the tools allows defining hierarchical systems, which combine fuzzy and nonfuzzy components, with complex rulebases and user-defined fuzzy operators. Verification and tuning tools permit to simulate the system behavior and to adjust the system parameters by means of different supervised learning algorithms. Software synthesis tools generate C or C++ codes that can be cross compiled and executed in the MicroBlaze processor, thus providing a fully software solution for embedded fuzzy controllers. Finally, the hardware synthesis tool included in the environment is able to generate a description of the system in VHSIC Hardware Description Language (VHDL) according to the optimized architecture for the realization of fuzzy systems previously described. When the designer runs the synthesis process, different architectural and implementation options can be chosen according to the particular characteristics of the problem. The generated VHDL description of the system can be further converted into an IP module to be used with the tools of Xilinx's EDK environment.

B. Using Fuzzy IP Modules in EDK

The EDK environment includes a set of IP components and tools that simplify the development of embedded processing systems on FPGAs. The Xilinx Platform Studio (XPS) graphical user interface provides a group of templates that facilitate the design of OPB-compatible peripherals. These templates include a VHDL code that describes two components: IP Interface ("IPIF"), which performs the interface functions with the OPB bus; and "User_logic," which includes the logic developed by the user. In our case, the latter is composed of a VHDL description of the FIM given by *Xfuzzy* plus the necessary code to access the controller through the registers of the IP module. Both components communicate through the IP Interconnect ("IPIC") interface, which is independent on the peripheral bus. Different types of templates exist depending on the operation mode of the peripheral (master/slave) and the services provided by the IPIF block. The XPS graphical user interface also facilitates the generation of files required for the use of the IP module as another peripheral of the system.

Fig. 5 shows the design flow that allows integrating the MicroBlaze processing system and the inference module in order to obtain a SoPC embedded fuzzy controller. Once the FIM is defined and verified, and after obtaining its VHDL code with the help of the synthesis tool of *Xfuzzy*, the FIM can be encapsulated as an OPB-compatible peripheral. Then, a typical development cycle for hybrid HW/SW systems can be carried out with Xilinx's EDK environment. MicroBlaze components (including the OPB-connected FIM) defined in the hardware specification file (".mhs" file) are synthesized and implemented by using EDK's *platgen* tool and Xilinx's ISE development environment. The obtained bit stream (".bit" file) will be used to configure the FPGA internal connections that allow determining its behavior. MicroBlaze drivers and libraries for the included peripherals are generated by using EDK's *libgen* tool according to the software specifications file (".mss" file). Application programs can be compiled and linked with the GNU cross-development tools also provided by the

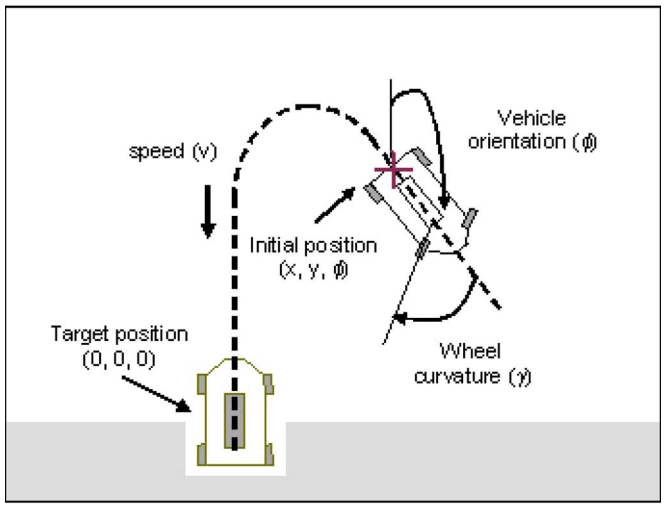

Fig. 6. Parking problem illustration.

EDK environment. Finally, the debugging tool *xmd* permits to download the executable code (".elf" file) into the program memory.

V. Fuzzy Control System for Autonomous Mobile Robots

The parking of autonomous vehicles in a constrained space is a typical control problem in robotics [28], [29]. Fig. 6 illustrates the diagonal parking problem addressed in this paper. Starting from any given position and orientation (x, y, ϕ), the autonomous mobile robot must drive forward and backward (as required) at speed v and with a wheel curvature γ in order to always arrive backward at target position (0, 0, 0).

The mobile robot used has been an autonomous electric vehicle called Romeo-4R [30]. Romeo-4R is a four-wheeled car with standard Ackerman steering, direct current traction and steering electrical motors, and a set of sensors and actuators that make it capable of autonomous navigation. A digital signal processor (DSP) TMS-320LF from Texas Instruments provides support for motor control (encoder inputs and pulsewidth-modulated outputs), analog-to-digital conversion, and communication links through serial ports, thus easing the low-level control of the vehicle [31]. The DSP acquires information from sensors (a gyroscope and encoders) and processes it by using a kinematical model usually employed for car-like robots in order to resolve the actual position (x, y) and orientation (ϕ) according to

$$\begin{aligned} \dot{x} &= v \cdot \sin(\phi) \\ \dot{y} &= v \cdot \cos(\phi) \\ \dot{\phi} &= \gamma \cdot v. \end{aligned} \tag{2}$$

Once the values of the current state (x, y, ϕ, v, and γ) of Romeo-4R are known, the DSP transmits them to the FPGA containing the fuzzy controller through an RS-232 serial interface and using a specific communication protocol (which is also implemented by the program running in the MicroBlaze processor). The state of the vehicle is transmitted every 50 ms, thus determining the duration of the control cycle. The fuzzy high-level controller performs the parking control strategy and sends back to the DSP the new required values of speed and wheel curvature so that the DSP controls the traction and direction motors. This hierarchical control structure allows developing different control strategies in the high-level controller and frees it from the low-level control task of Romeo-4R. As corresponds to a reactive approach, the fuzzy controller responds to the present inputs using a stimulus/response type of behavior that does not take into account the past history of the vehicle. For this reason, possible errors in the calculus of the vehicle's state due to road friction, sensor inexactitudes, etc., only affect the current iteration and may be compensated in the next control cycles.

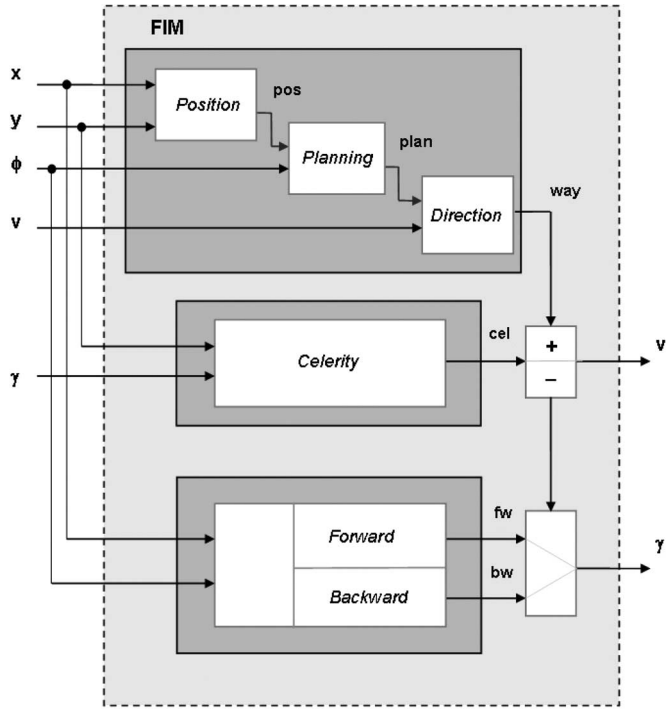

Fig. 7. FIM structure for parking control.

Following the proposed hybrid HW/SW strategy, the MicroBlaze processor performs the interface protocol to communicate with the low-level controller through an OPB-connected UART module, the sequencing of the different processing stages, and the information interchange with the FIM. This FIM implements a motion planning strategy based on rules that emulate the expert knowledge of a human driver. Its design process has been carried out with the help of the *Xfuzzy* environment [32]. Different FIMs have been tested: from simple modules aimed at implementing simple behaviors to the hierarchical structure shown in Fig. 7 containing six rulebases with multiple inputs and outputs.

The rulebases *Position*, *Planning*, and *Direction* in Fig. 7 correspond to decision-making rulebases that provide as outputs the labels that represent the different situations of the vehicle. On the contrary, *Celerity*, *Forward*, and *Backward* are interpolating rulebases that provide numeric output values. *Position* determines if the vehicle is near or far from the target

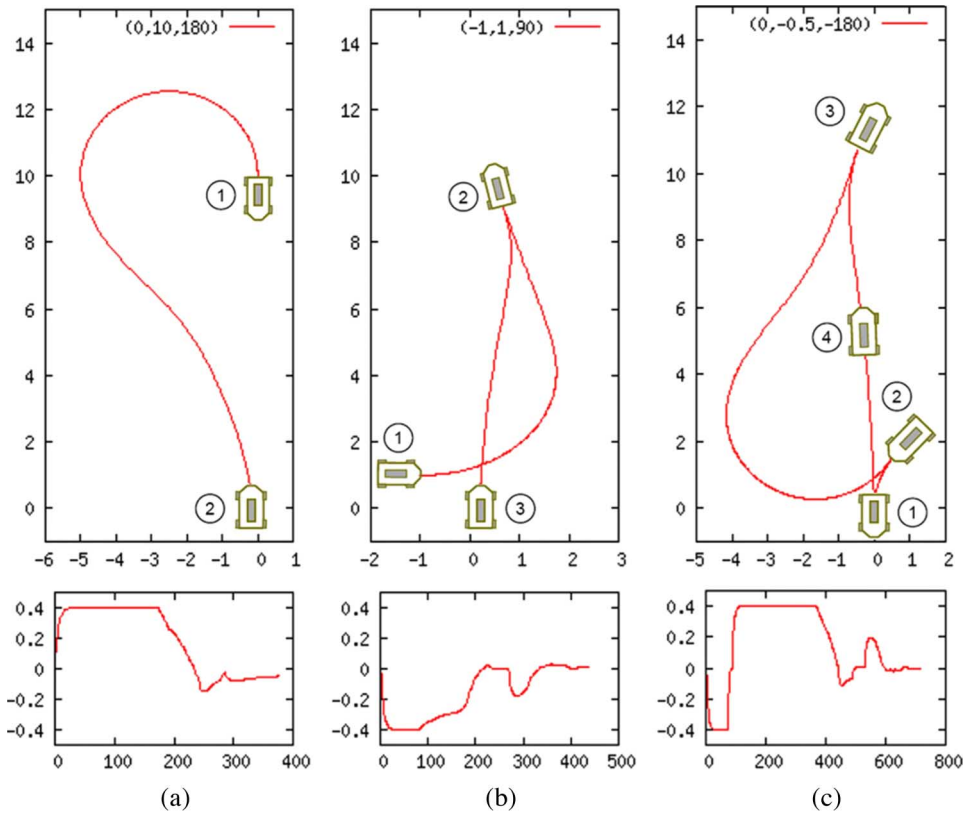

Fig. 8. Experimental results of parking maneuvers from different initial positions.

position and if it is centered or not with respect to this position. Its output (*pos*) is combined with the vehicle orientation in the *Planning* rulebase to propose a driving direction (*plan*). Afterward, the plan output is combined with the previous speed value at the *Direction* rulebase to obtain the actual driving direction (*way*). *Celerity* determines the absolute value (*cel*) of the vehicle speed. The heuristic implemented by this rulebase is simple: the vehicle will drive slower when the absolute value of its curvature is big and when the car is arriving at the parking place. *Backward* and *Forward* set the vehicle curvature. The parameters defining the behavior of these rulebases have been tuned using the supervised learning tools provided by *Xfuzzy* and a training file with data obtained from the geometrical consideration of the problem taking into account the nonholonomic constraints of Romeo-4R. Both rulebases share the input variables, which makes it possible to implement them in a single module. Finally, references to the traction and direction engines (new values of v and γ) can be obtained as a function of *way* by means of a sign selector and a multiplexer, as illustrated in Fig. 7.

VI. Experimental Results

Once the FIM is implemented and encapsulated as an IP module, it has been combined with an adequate configuration of the MicroBlaze processing system in order to build the fuzzy controller. The hardware realization of this controller consumes 3438 Slices of the FPGA (approximately 45% of the Spartan-xc3s1000 available in the development board). The 60% of these resources are required to implement the MicroBlaze processing system and its associated components. The remaining 40% correspond to the FIM. The controller also employs 8 of the 24 hardware multipliers available in the FPGA (three for the MicroBlaze core, one for the *Celerity* rulebase, and four for the combined *Backward–Forward* rulebase) and 16 RAM memory blocks that provide 32 KB of local memory used to execute the control programs.

The high-level C program running in the MicroBlaze processor is in charge of the following tasks: receiving the status information coming from the DSP, performing the required data-type conversions, writing the inputs to the FIM, reading the corresponding outputs, calculating the new values of speed and curvature, and sending them to the DSP. The control continues until the vehicle has arrived at the target position, a predefined number of iterations has been reached, or more than three transmission errors have been detected.

Both the MicroBlaze system and the FIM operate with the 50-MHz clock signal available in the development board. All the operations carried out by MicroBlaze in each iteration are performed in less than 3 μs, which is far from the 50-ms control cycle fixed by the DSP-based low-level controller. Once its inputs are established, the FIM completes the inference process and provides a valid output after 16 clock cycles (320 ns).

The upper graphs in Fig. 8 show experimental results illustrating the trajectories of several parking maneuvers that start from different initial positions. The lower graphs in Fig. 8 show the temporal evolution of the vehicle's curvature (γ) in the three considered situations. Fig. 8(a) corresponds to a case where the vehicle faces the parking place, and it is far enough so as to be able to reach the target by driving only backward.

Romeo-4R starts describing an arc of maximum curvature until its orientation is near 45° and then changes the sign of the curvature and progressively reduces its value to approach the parking position. The trajectory in Fig. 8(b) corresponds to a very usual case where the initial orientation of the vehicle is perpendicular to the parking place. Romeo-4R is driven forward by the fuzzy controller first with maximum curvature and then with smaller values in order to situate the robot near the $x = 0$ axis and with orientation $\phi = 0$. From this point, the vehicle is driven backward toward the parking site. Finally, a more complex trajectory is illustrated in Fig. 8(c). In this case, Romeo-4R is initially parked at the target position but with a wrong orientation. To correct this situation, the vehicle is first driven backward to leave the parking place and then follows a path similar to the one described in case b. As can be observed in the graphs, all the transitions are smooth when the robot is moving backward or forward.

VII. Conclusion

A realization strategy for the development of hybrid HW/SW embedded fuzzy controllers on FPGA devices has been described. The main characteristic of this strategy is the use of IP modules for the realization of both the general-purpose processing system that carries out the software-assigned task and the fuzzy modules implemented by specific hardware that accelerates the inference processes. The combination of this strategy with a set of generic and specific design tools allows the rapid development of fuzzy controllers with excellent flexibility and performance characteristics. In addition, the use of modern logical programmable devices for hardware support provides rapid and low-cost implementations for consumer products that do not require high production volumes. In order to validate this approach, a configurable platform for embedded fuzzy controllers has been presented and used to develop a fuzzy control system for the diagonal parking of an electrical vehicle. The obtained results demonstrate the capabilities of the fuzzy logic techniques to cope with these types of problems, and the utility of the platform and its associated chain tool to confront the design of complex fuzzy controllers.

Acknowledgment

The authors would like to thank the Department of "Ingeniería de Sistemas y Automática", University of Seville, in particular, V. Blanco and J. Ferruz, for advice and collaboration to carry out the experimental tasks with Romeo-4R.

References

[1] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 1, pp. 28–44, Jan. 1973.

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

[3] L. Reyneri, "Implementation issues of neuro-fuzzy hardware: Going toward HW/SW codesign," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 176–194, Jan. 2003.

[4] W. Savage, J. Chilton, and R. Camposano, "IP reuse in the system on a chip era," in *Proc. Int. Symp. Syst. Synthesis*, Sep. 2000, pp. 2–7.

[5] J. Becker, "Configurable systems-on-chip (CSoC)," in *Proc. Symp. Integr. Circuits and Syst. Design*, Sep. 2002, pp. 379–384.

[6] A. Costa, A. De Gloria, F. Giudici, and M. Olivieri, "Fuzzy logic microcontroller," *IEEE Micro*, vol. 17, no. 1, pp. 66–74, Jan./Feb. 1997.

[7] V. Salapura, "A fuzzy RISC processor," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 6, pp. 781–790, Dec. 2000.

[8] R. Bannatyne, "Motorola's 68HC12 an evolution from 8-bit to 16-bit," *Embedded Syst. Eng.*, vol. 4, no. 4, pp. 32–33, Jun./Jul. 1996.

[9] L. Fortuna, M. Lo Presti, C. Vinci, and A. Cucuccio, "Recent trends in fuzzy control of electrical drives: An industry point of view," in *Proc. Int. Symp. Circuits and Syst.*, May 2003, vol. 3, pp. 459–461.

[10] D. Hung, "Dedicated digital fuzzy hardware," *IEEE Micro*, vol. 15, no. 4, pp. 31–39, Aug. 1995.

[11] M. Togai and H. A. Watanabe, "A VLSI implementation of a fuzzy inference engine: Toward an expert system on a chip," *Inf. Sci.*, vol. 38, no. 2, pp. 147–163, Apr. 1986.

[12] T. Yamakawa and T. Miki, "The current mode fuzzy logic integrated circuits fabricated by the standard CMOS process," *IEEE Trans. Comput.*, vol. 35, no. 2, pp. 161–167, Feb. 1986.

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

[14] H. Eichfeld, T. Kunemund, and M. Menke, "A 12b general purpose fuzzy logic controller chip," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 4, pp. 460–475, Nov. 1996.

[15] K. Shimizu, M. Osumi, and F. Imae, "Digital fuzzy processor FP-5000," in *Proc. Int. Conf. Fuzzy Logic & Neural Netw.*, Jul. 1992, pp. 539–542.

[16] A. Pagni, "Digital approaches," in *Handbook of Fuzzy Computation*. Bristol, PA: Inst. of Phys., 1998.

[17] D. Kim, "An implementation of fuzzy logic controller on the reconfigurable FPGA system," *IEEE Trans. Ind. Electron.*, vol. 47, no. 3, pp. 703–715, Jun. 2000.

[18] M. McKenna and B. M. Wilamowski, "Implementing a fuzzy system on a field programmable gate array," in *Proc. Int. Joint Conf. Neural Networks*, Jul. 2001, vol. 1, pp. 189–194.

[19] C.-F. Juang and J.-S. Chen, "Water bath temperature control by a recurrent fuzzy controller and its FPGA implementation," *IEEE Trans. Ind. Electron.*, vol. 53, no. 3, pp. 941–949, Jun. 2006.

[20] A. Barriga, R. Senhadji, C. J. Jiménez, I. Baturone, and S. Sánchez-Solano, "A design methodology for application specific fuzzy integrated circuits," in *Proc. IEEE Int. Conf. Electron., Circuits and Syst.*, Sep. 1998, vol. 1, pp. 431–434.

[21] B. M. Wilamowski, R. C. Jaeger, and M. O. Kaynak, "Neuro-fuzzy architecture for CMOS implementation," *IEEE Trans. Ind. Electron.*, vol. 46, no. 6, pp. 1132–1136, Dec. 1999.

[22] A. Cabrera, S. Sánchez-Solano, R. Senhadji, A. Barriga, and C. J. Jiménez, "Hardware/software codesign methodology for fuzzy controllers implementation," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, May 2002, pp. 464–469.

[23] A. Cabrera, S. Sánchez-Solano, P. Brox, A. Barriga, and R. Senhadji, "Hardware/software codesign of configurable fuzzy control systems," *Appl. Soft Comput.*, vol. 4, no. 3, pp. 271–285, Aug. 2004.

[24] *MicroBlaze Reference Guides*, Xilinx, Inc., San Jose, CA, Aug. 2006.

[25] S. Sánchez-Solano, A. Barriga, C. J. Jiménez, and J. L. Huertas, "Design and applications of digital fuzzy controllers," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 1997, vol. 2, pp. 869–874.

[26] F. J. Moreno-Velo, I. Baturone, S. Sánchez-Solano, and A. Barriga, "Rapid design of fuzzy systems with Xfuzzy," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, May 2003, pp. 342–347.

[27] *Xfuzzy: Fuzzy Logic Design Tools*. [Online]. Available: http://www.imse.cnm.es/Xfuzzy

[28] T.-H. S. Li, C. Shih-Jie, and C. Yi-Xiang, "Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot," *IEEE Trans. Ind. Electron.*, vol. 50, no. 5, pp. 867–880, Oct. 2003.

[29] C.-L. Hwang, L.-J. Chang, and Y.-S. Yu, "Network-based fuzzy decentralized sliding-mode control for car-like mobile robots," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 574–585, Feb. 2007.

[30] F. Cuesta, F. Gómez-Bravo, and A. Ollero, "Parking maneuvers of industrial-like electrical vehicles with and without trailer," *IEEE Trans. Ind. Electron.*, vol. 51, no. 2, pp. 257–269, Apr. 2004.

[31] J. Ferruz, V. Blanco, A. Ollero, and J. V. Acevedo, "An embedded DSP-based controller for the Romeo-4R vehicle," in *Proc. IFAC Int. Symp. Intell. Compon. and Instrum. Control Appl.*, Jul. 2003, pp. 101–106.

[32] I. Baturone, F. J. Moreno-Velo, S. Sánchez-Solano, and A. Ollero, "Automatic design of fuzzy controllers for car-like autonomous robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 447–465, Aug. 2004.

Santiago Sánchez-Solano received the Licenciado en Físicas degree (with honors) and the Ph.D. degree in physics from the University of Seville, Seville, Spain, in 1980 and 1990, respectively.

After six years as a System Analyst at the Computer Center, University of Seville, he joined the Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, CNM-CSIC, Seville, where he is currently a Senior Researcher. His main research interests include VLSI design, CAD tools for microelectronic design, and hardware implementation of neuro-fuzzy systems.

Alejandro J. Cabrera received the Diploma in electronic engineering, the M.S. degree in digital systems, and the Ph.D. degree in technical sciences from the Instituto Superior Politécnico José Antonio Echeverría (CUJAE), Havana, Cuba, in 1977, 1985, and 2004, respectively.

Since 1977, he has been with CUJAE, where he is currently a Titular Professor of electronics and digital systems in the Department of Automation and Computers. His research interests include the development of embedded systems and hybrid hardware–software implementation of fuzzy control systems.

Iluminada Baturone received the Licenciado en Físicas degree (with honors) and the Ph.D. degree in physics (with honors) from the University of Seville, Seville, Spain, in 1991 and 1996, respectively.

Since 1990, she has been with the Instituto de Microelectrónica de Sevilla, Centro Nacional de Microelectrónica, CNM-CSIC, Seville, with undergraduate, postgraduate, and postdoctoral fellowships. She is currently an Associate Professor in the Department of Electronics and Electromagnetism, University of Seville. Her current research interests include hardware implementation and CAD tools for neuro-fuzzy systems, and robotics and image processing applications.

Francisco J. Moreno-Velo received the Licenciado en Físicas degree and the Licenciado 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 in the Department of Applied Physics and Electrical Engineering, University of Huelva, Huelva, Spain. From 2000 to 2004, he was a Researcher at the Instituto de Microelectrónica de Sevilla, CNM-CSIC, Seville. Since 2004, he has been an Assistant Professor in the Department of Information Technologies, University of Huelva. His research interests include soft-computing techniques, CAD tools for fuzzy systems, and compiler design.

María Brox received the Licenciado en Físicas degree (with honors) from the University of Córdoba, Córdoba, Spain, in 2004.

From 2005 to 2007, she had a postgraduate fellowship from the Spanish government in the Instituto de Microelectrónica de Sevilla, CNM-CSIC, Seville, Spain. She is currently an Assistant Professor in the Department of Electronics, University of Córdoba. Her research area is the development of IP modules of fuzzy controllers for the design of embedded systems on FPGAs.