# Model-Based Design Methodology for Rapid Development of Fuzzy Controllers on FPGAs

Santiago Sánchez-Solano, María Brox, Ernesto del Toro, Piedad Brox, and Iluminada Baturone

***Abstract*—The complexity reached by current applications of industrial control systems has motivated the development of new computational paradigms, as well as the employment of hybrid implementation techniques that combine hardware and software components to fulfill systems requirements. On the other hand, continuous improvements in field programmable devices make today possible the implementation of complex control systems on reconfigurable hardware, although they are limited by the lack of specific design tools and methodologies to facilitate the development of new products. This paper describes a model-based design approach for the synthesis of embedded fuzzy controllers on FPGAs. Its main contributions are the proposal of a novel implementation technique, which allows accelerating the exploration of the design space of fuzzy inference modules, and the use of a design flow that eases their integration into complex control systems and the joint development of hardware and software components. This design flow is supported by specific tools for fuzzy systems development and standard FPGA synthesis and implementation tools, which use the modeling and simulation facilities provided by the Matlab environment. The development of a complex control system for parking an autonomous vehicle demonstrates the capabilities of the proposed procedure to dramatically speed up the stages of description, synthesis, and functional verification of embedded fuzzy controllers for industrial applications.**

***Index Terms*—Fuzzy controllers, Hardware/Software codesign, Industrial control systems, Model-based design.**

## I. Introduction

THE rapid evolution of microelectronics market causes a continuous increment in the demand of new products with higher levels of functionality and performance. In addition to the constant improvements of integrated circuits technologies, to accomplish these requirements new computational paradigms (like Soft-Computing) have been proposed and developed in the last years [1]-[3]. In particular, the capability of fuzzy logic–based inference systems to describe complex control strategies by means of linguistic rules, avoiding the need for mathematical models and providing good results in terms of adaptability and robustness, has motivated an increasing interest for the use of this technique to implement intelligent control systems in applications related to industrial control, robotics, and consumer electronics [4]-[6].

As a consequence, different implementation approaches for fuzzy systems have been proposed in the literature, which range from software implementation using computer programs to microelectronic realization by means of application specific integrated circuits (ASIC) or programmable devices [7]-[8]. Recent advances in FPGA technologies have promoted a boom of this kind of devices, as current FPGA families provide enough resources as to allow the implementation of a complete system on a programmable chip (SoPC) [9]-[25].

On the other hand, the level of complexity reached by current applications of industrial control systems, as well as their requirements of speed, size, and/or power consumption, makes fuzzy inference modules (FIM) to be conceived as components of a whole system, which usually includes different hardware (HW) and software (SW) components to carry out its function [19]-[25]. In order to reduce the development cycle of new products and make them more competitive in market terms, the design of these hybrid HW/SW systems requires the use of new methodologies and design tools to facilitate the concurrent development of their different components. A design strategy that allows the rapid development of hybrid fuzzy controllers with adequate characteristics of flexibility and performance was introduced by the authors in [19].

Advancing in that research line, this paper describes a model-based codesign technique for the synthesis of embedded fuzzy control systems on FPGAs. The proposed design flow combines specific fuzzy system development tools from the *Xfuzzy* environment [26], FPGA synthesis and implementation tools from Xilinx, and modeling and simulation tools from Matlab. The first advantage of this approach is that it allows accelerating the exploration of the design space of inference modules to be included in fuzzy control systems. In addition, it facilitates their integration as peripheral of a general purpose

S. Sánchez-Solano is with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain (phone: +34 954466666; fax: +34 954466600; e-mail: santiago@imse-cnm.csic.es).

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

I. Baturone is with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, Spain, and also with the University of Seville, Seville, Spain (email: lumi@imse-cnm.csic.es).

processor and makes possible the concurrent synthesis and verification of hardware and software components in the system.

The structure of the paper is the following. Key concepts regarding FPGA implementation of embedded fuzzy control systems are introduced in Section II. The main contribution of this work is presented in Section III, where a novel implementation technique for fuzzy systems is described. It is based on the use of digital signal processing (DSP) design tools in combination with a cell library developed by the authors for efficient implementation of fuzzy inference systems. The design flow associated to the proposed technique is detailed in Section IV. The implementation of a complex control system for parking an autonomous vehicle is illustrated in Section V. Finally, Section VI summarizes the main conclusions of this work.

## II. FPGA Implementation of Embedded Fuzzy Controllers

The continuous evolution of programmable logic devices has encouraged the use of FPGAs both as prototyping and final platforms of industrial control systems (as the Takagi–Sugeno recurrent fuzzy network used in [12] for water bath temperature control, the adaptive fuzzy controller for a permanent-magnet linear synchronous motor drive system presented in [13], or the fuzzy logic controller for boost converter output-voltage regulation proposed in [14]). Many of these FPGA implementations of fuzzy controllers are used in robotics applications to control the maneuvers of mobile robots [15]-[17]. Different solutions have been also proposed to cope with the increasing complexity of embedded control systems. The system described in [18] takes advantage of the run-time reconfiguration facility of modern FPGAs. Other authors resort to hybrid HW/SW approaches using the soft cores available in Xilinx's [19]-[20] or Altera's [21]-[24] FPGAs.

According to the HW/SW strategy for implementation of fuzzy controllers proposed in [25], an efficient solution for complex control systems consists of using a general purpose processor (to carry out initialization, communication, and global control tasks) in combination with dedicated hardware (to accelerate the fuzzy inference process). The design approach described in this paper uses the MicroBlaze processor, available as an Intellectual Property (IP) module for Xilinx's FPGAs, in combination with application specific FIMs, also developed as IPs, to speed up the inference tasks.

MicroBlaze is a 32-bit RISC processor available as a soft core optimized for implementation in Xilinx's FPGAs. It is based on a Harvard architecture that provides a Local Memory Bus (LMB), for connections of local memory, and a Processor Local Bus (PLB), for peripherals. There are many IP-modules of configurable peripherals compatible with this standard, which allow configuring systems according to specific applications. Design tools included in Xilinx Platform Studio (XPS) provide software drivers that ease the use of MicroBlaze peripherals, as well as facilities for converting custom hardware into PLB-compatible IP-modules [27].

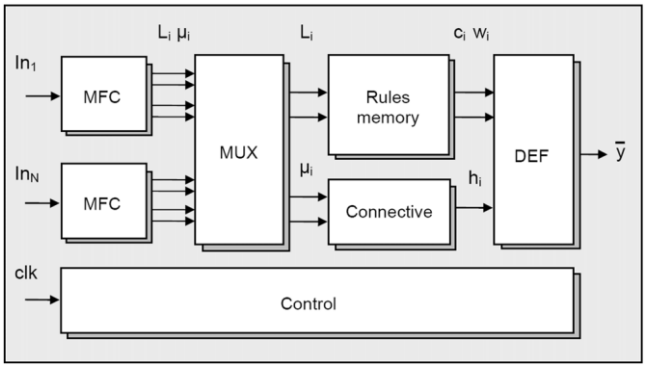



Fig.1. Active-rule based architecture for efficient implementation of digital fuzzy controllers [19].

Specific FIMs required by fuzzy control systems may be implemented on Xilinx's FPGAs as PLB-compatible peripherals of the MicroBlaze processor. In order to obtain an efficient realization of these modules, the active-rule based architecture for fuzzy inference systems described in [19] is used to implement the FIMs. This architecture allows an efficient implementation of digital fuzzy system in terms of use of resources, power, and speed. These characteristics are accomplished by using a limited overlap degree of input membership functions, implementing a processing strategy that evaluates only the active rules, and employing simplified defuzzification methods. The block diagram of the architecture used for FIM implementation is shown in Fig. 1.

Fuzzy logic can be considered nowadays as a mature technology. However the number of design environments and CAD tools that are able to translate the high-level description of a fuzzy system into an efficient hardware implementation is small. The problem is even worst for HW/SW realizations, because new specification, design, and verification methods are needed to handle these cases where different languages and design techniques are used within the same design. In order to deal with this handicap, the Matlab/Simulink environment has been used in this work because it provides a high-level software platform for scientific computing and data visualization, in addition to a potent interactive programming environment. The main advantage of this model-based approach is that designs can be defined, optimized, tested, and debugged quickly [28]-[31].

## III. Synthesis of Fuzzy Inference Modules with DSP Tools

FPGA manufactures have developed different design tools to ease the implementation of digital signal processing (DSP) algorithms on FPGAs. One of these facilities is the System Generator tool (SysGen) from Xilinx [32]. It is based on Simulink, the interactive tool for modeling, analysis, and simulation of dynamic systems integrated in Matlab. SysGen includes a Simulink library, named "Xilinx Blockset", which

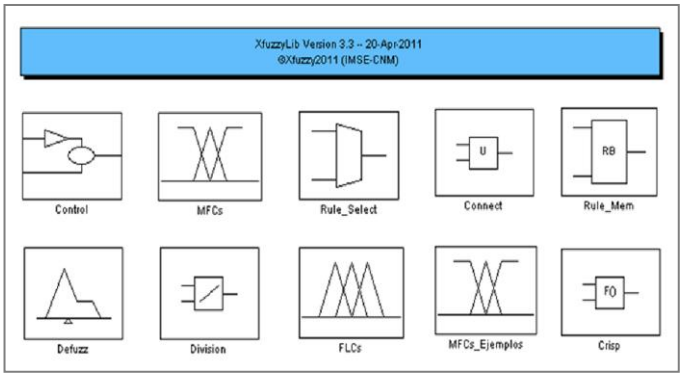



Fig.2. *XfuzzyLib* components grouped by functionalities.

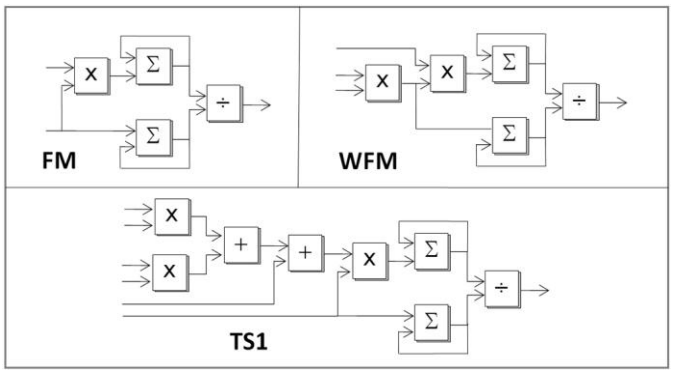



Fig.4. Simplified block diagrams of defuzzifiers included in *XfuzzyLib*.

provides basic building blocks for digital systems, as well as software components required to perform the synthesis and implementation on Xilinx's FPGAs of the algorithms described and simulated in Matlab.

The design flow with SysGen starts by describing the system schematic by means of a Simulink model. The verification stage is carried out by simulation, using the signal generation and visualization facilities offered by Matlab. Finally, the system implementation is performed by translating the model to different kinds of netlists and generating the bitstream file for the FPGA. When accomplishing the verification stage, SysGen is also able to include the appropriated interfaces to perform hardware co-simulation of the complete Simulink model. This simulation mode allows the combination of the hardware implementation of a part of the system on an FPGA development board in conjunction with the mathematical description of the rest of the model.

As a key element to support the design technique proposed in this paper, a cell library named *XfuzzyLib* has been developed using the SysGen tool. It includes different alternatives to implement each of the basic elements of the architecture for fuzzy controllers shown in Fig. 1. Fig 2 shows the components included in *XfuzzyLib* grouped by functionalities: membership function generators, active rule selectors, antecedent connectives, rule memories, defuzzifiers, and control elements. Membership function generators based on arithmetic techniques are able to generate families of membership functions corresponding to the types "triangular" and "sh_triangular" defined in *Xfuzzy* (all the elements of a sh_triangular family are functions with triangular shapes, except the first and the last elements that can be Z- and S-shaped functions, respectively). Fig. 3 shows the Simulink model of this component, which is built as an optimized and parameterized specific circuit using the basic building blocks provided by the Xilinx Blockset library.

Two options (product or minimum) for the connective used in the knowledge base of inference systems have been implemented in the library. Depending on the kind of fuzzy module being implemented, different defuzzifier blocks can be used: FuzzyMean (FM), WeightedFuzzyMean (WFM), or first-order Takagi-Sugeno (TS1), for interpolators; and MaxLabel, for decision-making systems [7]. Fig. 4 illustrates the simplified block diagrams of defuzzifiers available for the first type of fuzzy systems. Finally, 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.

As it happens to basic building blocks in "Xilinx Blockset", a set of parameters may be employed to define the size and functionality of *XfuzzyLib* components. When these

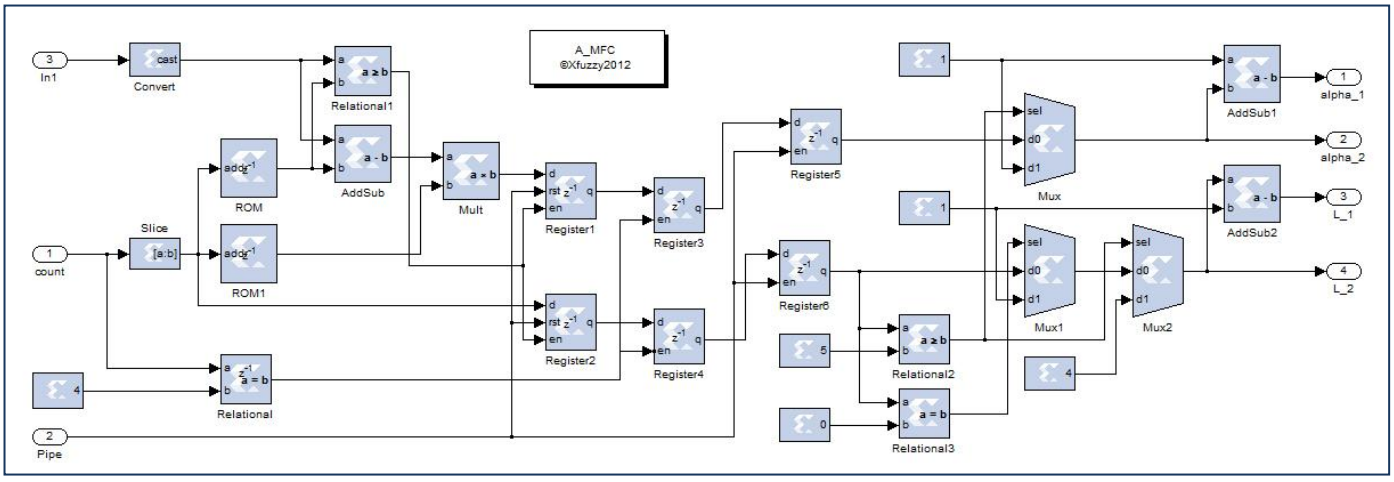



Fig. 3. Simulink model of the arithmetic membership function generator circuit (A_MFC) included in *XfuzzyLib*.

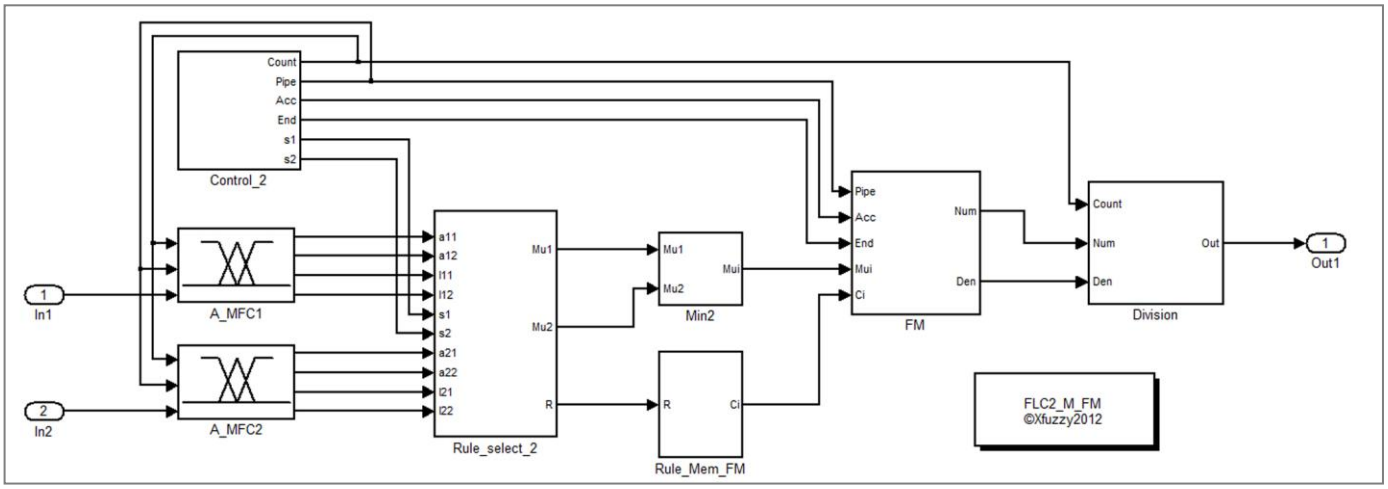



Fig. 5. Simulink model of a 2-imput 1-output FLC that uses minimum as connective and FuzzyMean defuzzification method.

components are instanced in a higher 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.

Building a fuzzy inference system with *XfuzzyLib* requires choosing, interconnecting, and defining the parameters of the needed blocks. As an example, Fig. 5 shows the Simulink model of a 2-input fuzzy module that uses the minimum operator as antecedent connective and employs FuzzyMean defuzzification method. 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.

*XfuzzyLib* also includes elements describing archetypal 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 the different defuzzification methods provided by the library. Blocks describing FLC architectures are fully parameterizable, making it possible to adapt its functionality according to the requirement of a particular application by defining the parameters related to the dimension of the system and the behavior of its knowledge database. In addition, the hierarchical combination of FLCs to define complex fuzzy systems is also possible.

## IV. HW/SW Codesign Approach

The design flow proposed in this work combines the use of specific tools for development of fuzzy systems from the *Xfuzzy* environment, FPGA synthesis and implementation tools provided by Xilinx, and simulation and modeling tools from Matlab. The development of a fuzzy control system will be carried out at the different stages illustrated in Fig. 6.

### A. FIM High-Level Design

The first stage of the design flow of an embedded fuzzy controller is about the description and functional verification of the fuzzy inference module (FIM). This task can be easy and rapidly accomplished using the tools included in the *Xfuzzy* design environment. *Xfuzzy* provides description,

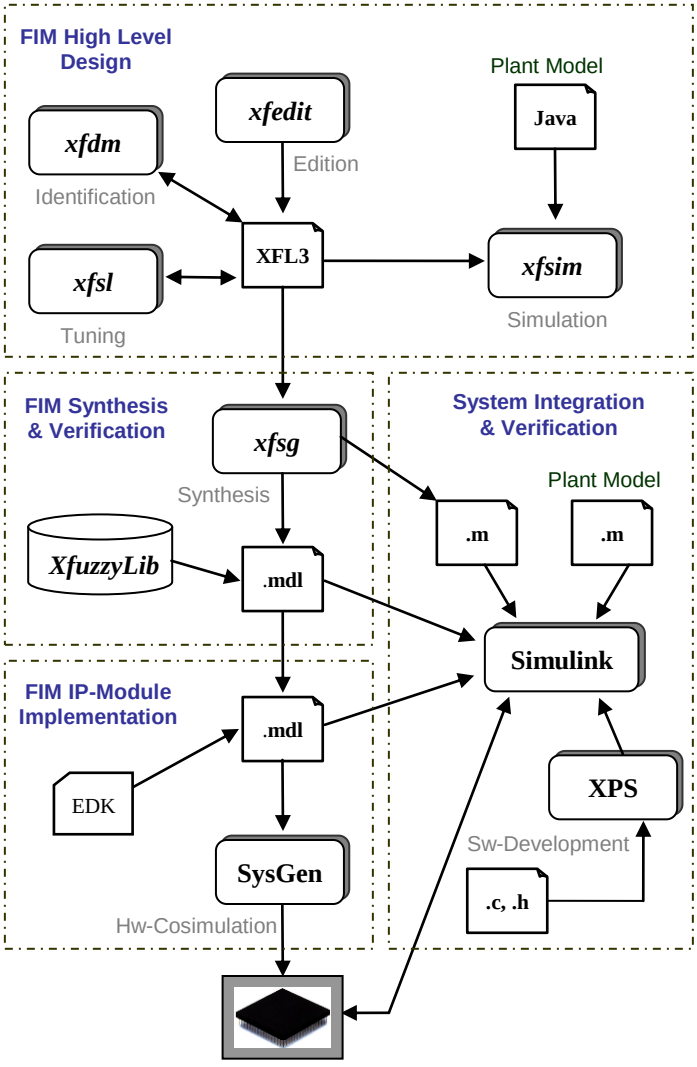



Fig. 6. Codesign approach for the synthesis of embedded fuzzy control systems on FPGAs.

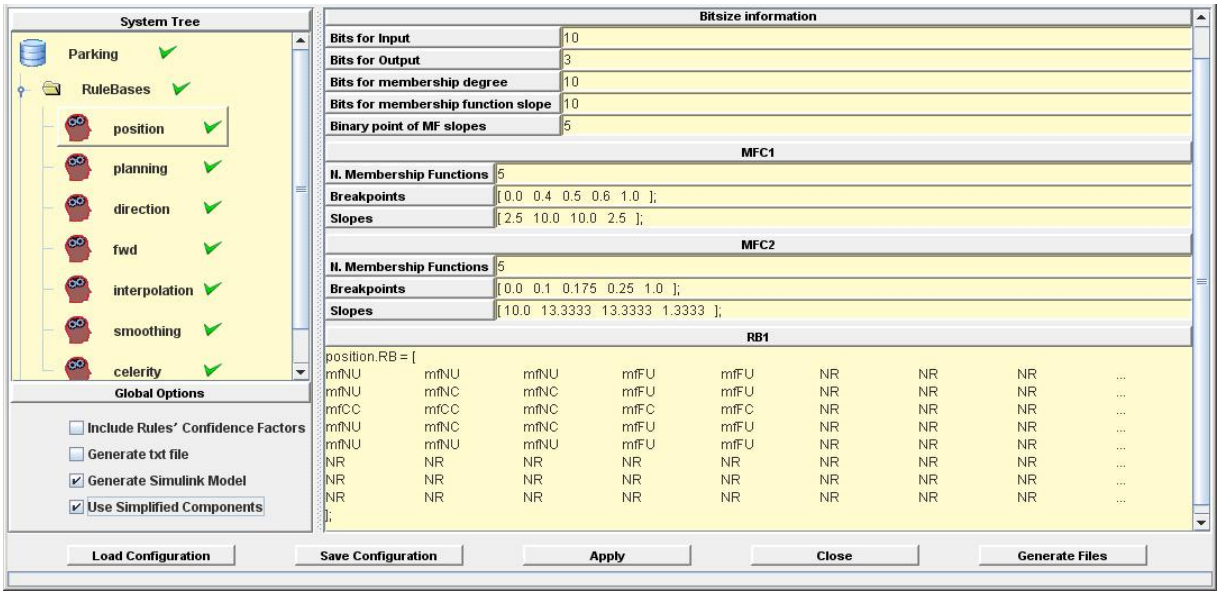



Fig. 7. Graphical user interface of the *xfsg* synthesis tool included in the *Xfuzzy* environment.

simplification, learning, and simulation tools, which allow defining and optimizing fuzzy inference modules, as well as synthesis tools to implement them as software or hardware components [26].

A fuzzy inference module is described in *Xfuzzy* by means of a specification in *Xfuzzy Specification Language* (XFL3), which can combine fuzzy blocks (to perform control and decision-making task) and crisp blocks (to implement general purpose arithmetic and logic functions). Knowledge rulebases can be directly defined by an expert operator (and edited with *xfedit*) or they can be extracted from numerical data using identification (*xfdm*) and supervised learning (*xfsl*) tools. Functional verification is carried out by analyzing the input-output relation of the system, as well as simulating its closed-loop behavior in combination with a Java-codified model of the plant (*xfsim*).

Hardware synthesis tools provided by *Xfuzzy* allow translating an XFL3 specification into a circuit description that can be implemented on a programmable device or an application specific integrated circuit [33]. To carry out the design approach described in this work, a new synthesis tool (*xfsg*), which is able to generate the files required by the codesign strategy, has been recently developed by the authors and incorporated to the *Xfuzzy* environment.

### B. FIM Synthesis and Verification

The *xfsg* synthesis tool generates the files required to synthesize FIMs according to the implementation technique described in Section III. The Graphical User Interface (GUI) of this tool is shown in Fig. 7. The upper left area contains the knowledge base, structured as a pull-down menu with components grouped under the categories "RuleBases" and "CrispBlocks". The right area gathers information about a particular rulebase or crisp block. Information related to membership functions and rules is directly extracted from the XFL3 specification, while data corresponding to size of buses are defined by means of parameters.

Once all the fuzzy system components have been configured, the tool allows generating the output files. Specifically, *xfsg* provides an ".mdl" file containing a Simulink model of the FIM, and an ".m" file with the parameters that define the size and functionality of FIM components.

### C. FIM IP-Module Implementation

The assembly of the FIM as IP-module and its integration with MicroBlaze processor must be performed at the next design stage. To accomplish this task, the input and output signals of the fuzzy controller are first connected to shared memory blocks (registers in this case) for communicating to MicroBlaze. Next, a MicroBlaze EDK processor block, available in Xilinx Blockset library, is added to the Simulink model. This block uses a MicroBlaze project previously generated from XPS.

During the import process, the shared memories are mapped onto MicroBlaze address space, so providing a mechanism to connect the FIM through a PLB interface. Basic drivers required to use the IP-module in software applications are automatically generated in this process.

### D. System Integration and Verification

According to the application needs, various forms of hardware synthesis and hardware co-simulation can be used to complete the final design stage. These options are all provided by the "System Generator" block. In particular, the use of hardware co-simulation facilities allows an FPGA implementation of the MicroBlaze processor to be simulated with different FIM peripherals connected through shared memory blocks in Matlab. This flexibility is a great advantage for verification and tuning the control system.

The development of hardware and software components is jointly performed at this stage using different tools and interfaces within Matlab environment. MicroBlaze software applications are coded and compiled with the help of XPS GUI, and are downloaded to the FPGA through the MicroBlaze block interface. In order to confirm the functionality of the control system into its real scenario, others blocks describing the plant and the operational environment should be also included in the Simulink model. Finally, blocks to initialize parameters, simulation control, and result visualization can be also incorporated to the model.

## V. Design Example

The above described design strategy has been applied to the realization of a fuzzy control system for autonomous parking of an electric vehicle. Romeo-4R is equipped with traction and steering motors, and different sensors that allow calculating position, orientation, and speed of the vehicle in every control cycle [34]. The goal of the control system is to act on the traction and steering motors to fix the adequate speed and wheel rotation angle so that the vehicle could park in a predefined position.

Romeo-4R uses a digital signal processor (DSP) TMS-320LF from Texas Instruments that acquires and processes information from sensors and provides support for motors control and communication links through a serial port, thus easing the low-level control of the vehicle (Fig. 8). The fuzzy control module interacts with the DSP using a communication protocol over RS232. The DSP sends to the fuzzy module the vehicle status information and, in response, transmits back to the vehicle the speed and wheel rotation angle calculated by the fuzzy controller.

In order to illustrate the design flow, this section describes the development of a hierarchical fuzzy control system able to implement a heuristic that allows parking the vehicle even when the initial position is near to the objective. This control system combines the strategies proposed in [19] (to park the vehicle using forward and backward maneuvers) and in [35] (to provide minimal-path in backward trajectories).

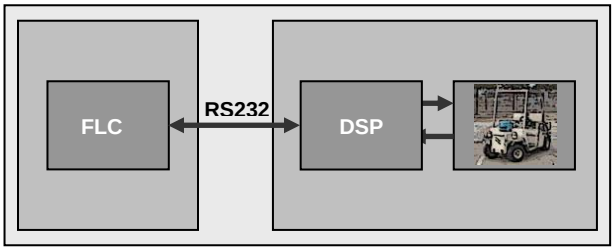
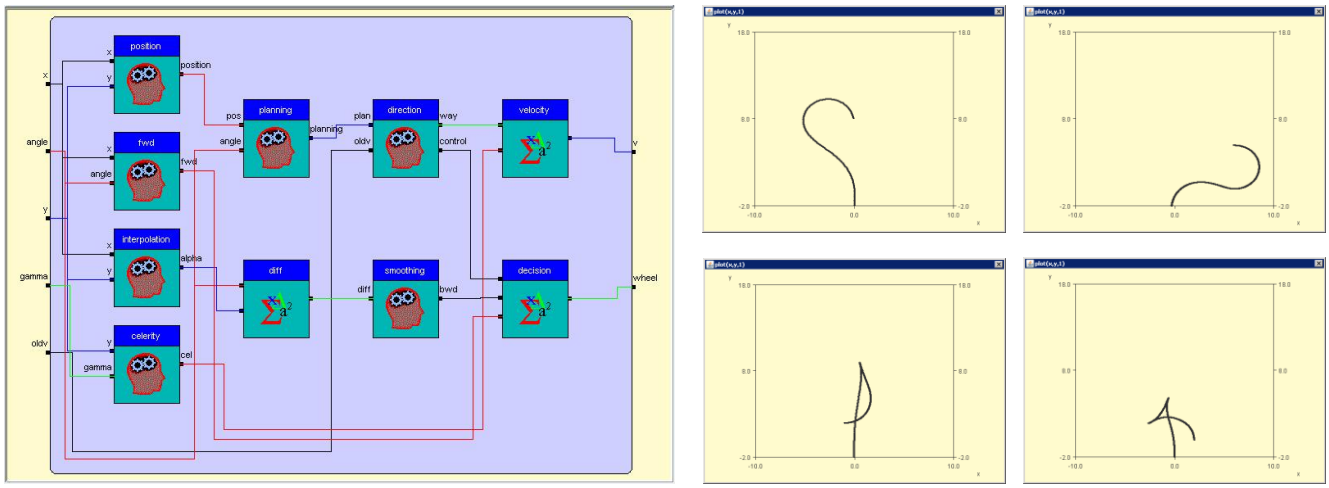



Fig. 8. Interaction between low- and high-level controllers in the Romeo-4R control system.

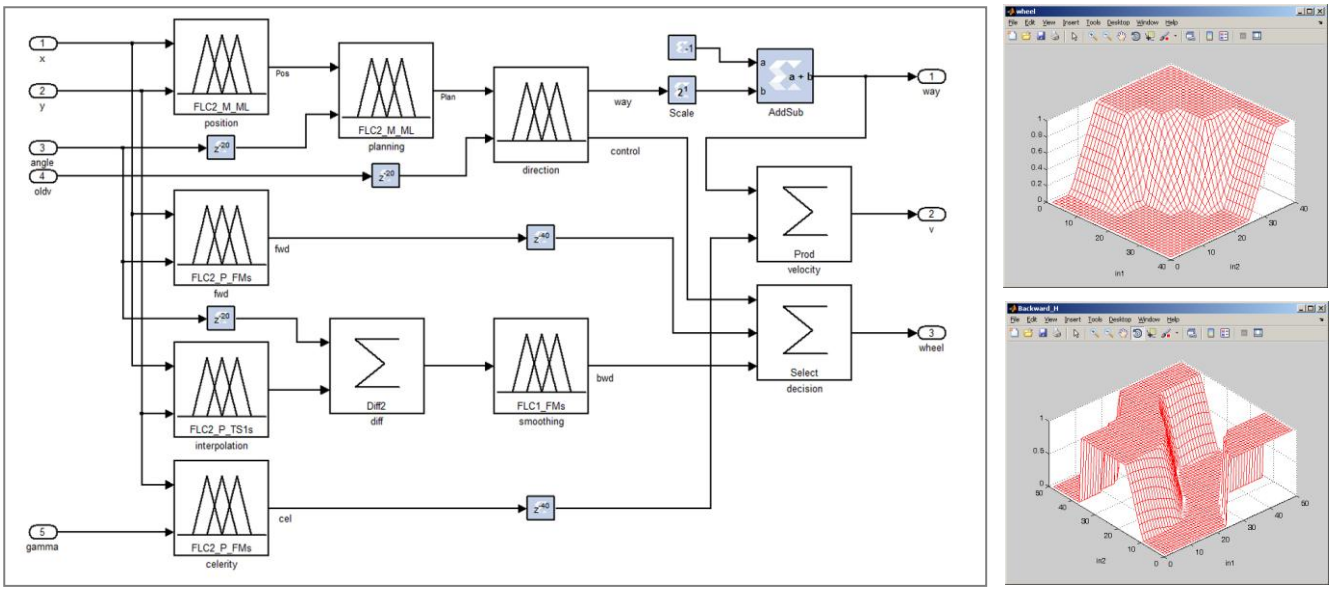
The structure of the FIM defined at the high-level design stage is shown in Fig. 9a. It is composed by seven rulebases and three crisp blocks. Rulebases called *Position*, *Planning*, and *Direction* perform a decision-making system that determines the direction of the vehicle. They use minimum as antecedent connectives and MaxLabel defuzzification method. *Celerity* and *Forward* are control rulebases that employ FuzzyMean defuzzifiers and provide, respectively, the absolute value of the vehicle speed and the wheel rotation angle for forward trajectories. Rulebase *Interpolation* is a first-order Takagi-Sugeno system that, in combination with *Smoothing* and *Diff* crisp block, implements the hierarchical fuzzy controller proposed in [35]. Finally, crisp blocks *Velocity* and *Decision* correspond to a multiplier and a multiplexer, respectively. Fig. 9b illustrates 1-, 2-, and 3-segment trajectories followed by Romeo-4R when it departs from different initial conditions.

Once the size of FIM components has been defined by means of the *xfsg* GUI, during the FIM Synthesis stage, the XFL3 specification is converted into the Simulink model




(a) (b)

Fig. 9. a) XFL3 specification of the FIM used for autonomous parking of Romeo-4R. b) Trajectories for different initial conditions of the vehicle.

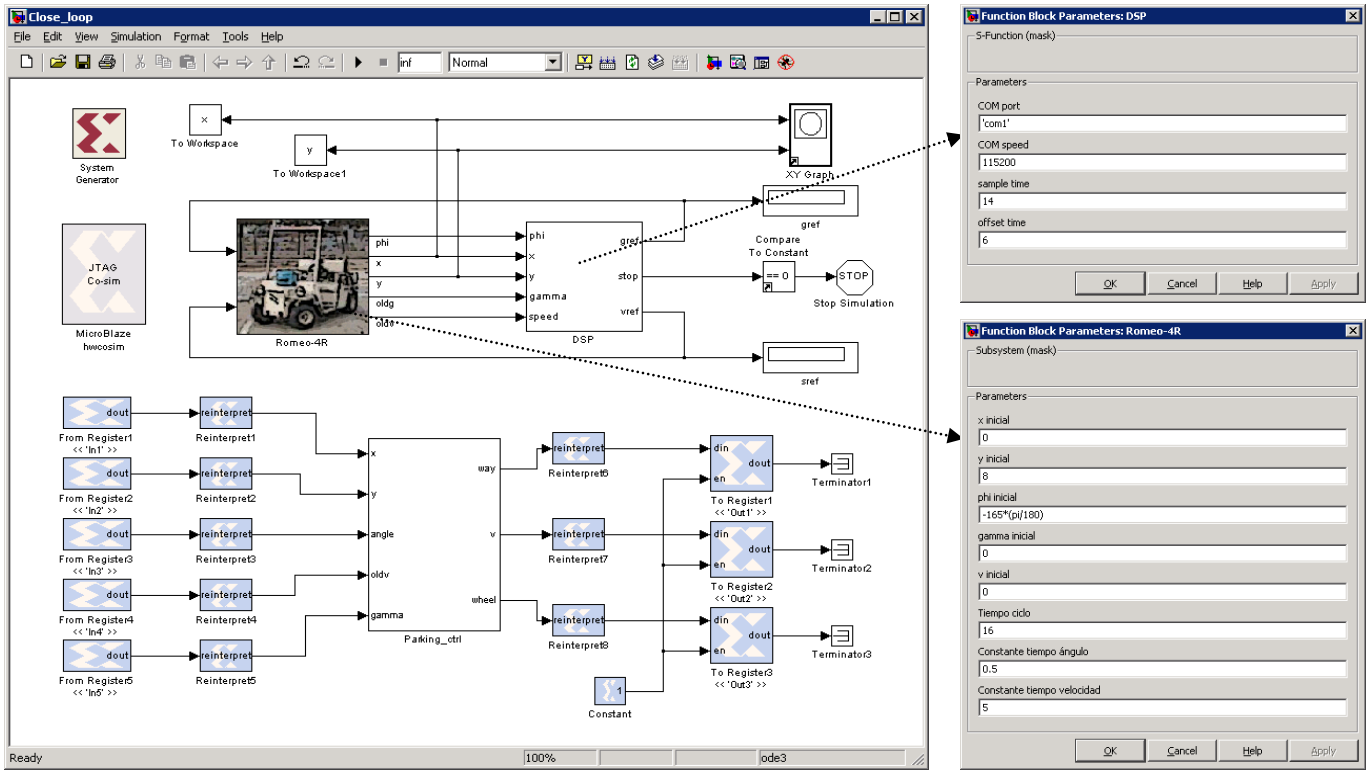



(a) (b)

Fig.10. a) Simulink model of the FIM generated by *xfsg*. b) Control surfaces for wheel angle in forward (top) and backward (down) maneuvers.

shown in Fig. 10a. FIM functionality is verified at this design phase using the simulation and graphical facilities provided by Matlab. As an example, Fig. 10b shows control surfaces that manage the wheel angle in forward and backward trajectories.

The objective of the next design stage is to convert the FIM into an IP-module that can be connected to the MicroBlaze processor as a standard PLB-peripheral. The first step to accomplish this task consists in associating shared memory blocks (from/to registers) to FIM's I/O ports, as illustrated in the bottom part of the Simulink model shown in Fig. 11a. Next, a MicroBlaze processor block, corresponding to a previously generated XPS project, is added to the Simulink




(a) (b)

Fig.11. Joint development and verification of hardware and software components using the proposed codesign methodology.

model. SysGen facilities allows implementing this subsystem as a hardware co-simulation block ("MicroBlaze hwcosim" block in Fig. 11a), which will be implemented in the FPGA and will interact with other model components while simulation is running. Using this strategy, different FIMs alternatives may be tested without having to re-implement the entire system, thus providing flexibility and ease of development.

In order to carry out the joint development of hardware and software components, during the system integration and verification stage, other blocks should be included in the Simulink model of Fig. 11a. "Romeo-4R" is a Simulink description of the vehicle dynamics based on a mathematical model, which allows verifying the system functionality in a closed control loop. The "DSP" block models the interaction between the low-level controller of Romeo-4R and the fuzzy controller under development. This model includes an RS232 link to communicate with the FPGA-implemented MicroBlaze processor by means of an ad-hoc communication protocol, which emulates the TMS-320LF functionality using a C++ Matlab S-Function. Finally, some extra blocks are used in the Simulink model of Fig. 11a to control the simulation and visualize the results. As shown in Fig. 11b, Matlab interfaces of the blocks "Romeo-4R" and "DSP" are used to set parameters that define the initial vehicle position and characteristics of the RS232 communication channel.

The communication protocol and the software drivers to communicate with the FIM peripheral, as well as the general control loop (coded in C and previously compiled) are executed by "MicroBlaze hwcosim" block implemented in the FPGA development board. Information between the hardware implemented controller and other components included in the Simulink model is transmitted through an RS232 cable connecting the UART incorporated in the MicroBlaze system and the serial port of the PC running Matlab. The FIM is able to complete an inference process after 10 clock cycles. This value is determined by the number of membership functions used by the *Smoothing* rulebase. Communication and control tasks carried out by MicroBlaze in each iteration require about 3 μs, while the control cycle fixed by the DSP device used by Romeo-4R is 50 ms.

Experimental results for different initial conditions obtained after implementing the embedded control system in an FPGA development board are shown in Fig. 12. As can be observed, these results are similar to simulation results provided by *Xfuzzy* in the FIM high-level design stage (Fig. 9b). Table I

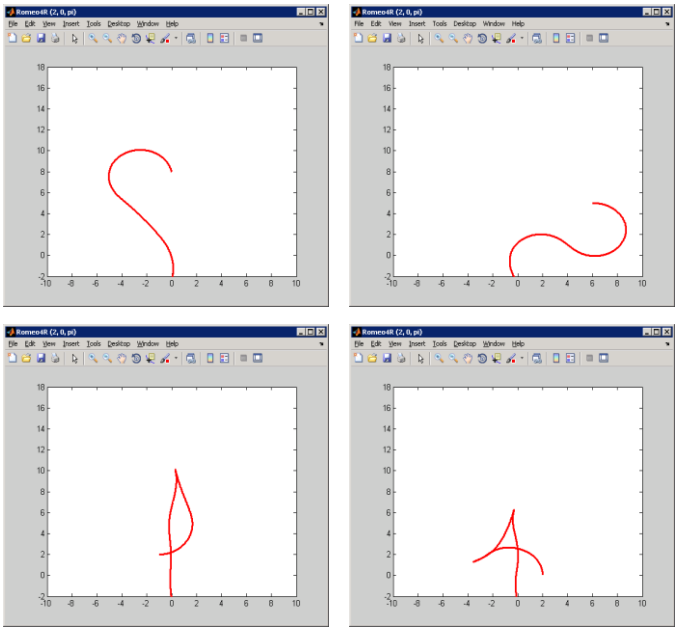


Fig. 12. Hardware co-simulation results for different initial conditions.

compares FPGA resource utilization and operation speed for the synthesis of the fuzzy controller, with different bit sizes, on a Spartan 3A DSP FPGA from Xilinx. Ten, twelve, fourteen, and sixteen bits for input and output precision are used in all the rulebases (except for output of decision-making ones, where only two or three bits are required to codify the different situations). The results, both in terms of resource utilization and operation speed, obtained by using the design technique proposed in this work are similar to those previously obtained using approaches based on VHDL descriptions [19]. However, the time required to carry out the system development is now drastically reduced. The fully-automated design flow based on SysGen allows implementing a FIM like the one used in this example in 10 or 15 minutes on a conventional workstation. The time required to complete the same task with the tools employed in [19] was at least an order of magnitude higher due to lacking of mechanisms for direct synthesis of hierarchical fuzzy modules.

This reduction in development time can be exploited, not only to accelerate the exploration of the design space of a given solution, but also to compare fuzzy controllers with different heuristic strategies. As an example, Table II shows implementation results for four fuzzy controllers of different complexity. Case 1 corresponds to the simplest solution that

TABLE I
IMPLEMENTATION RESULTS FOR DIFFERENT BIT-SIZE

| | 10-bit | 12-bit | 14-bit | 16-bit |
|---|---|---|---|---|
| *I/O Ports* | 73 (14%) | 87 (16%) | 101 (19%) | 115 (22%) |
| *Slices* | 1413 (8%) | 1510 (9%) | 1648 (10%) | 1989 (11%) |
| *DSP48As* | 23 (27%) | 23 (27%) | 23 (27%) | 23 (27%) |
| *RAM Blocks* | 10 (11%) | 10 (11%) | 10 (11%) | 10 (11%) |
| *Max. Freq. (MHz)* | 50.940 | 46.260 | 47,181 | 50,431 |

TABLE II
IMPLEMENTATION RESULTS FOR DIFFERENT FUZZY CONTROLLERS

| | Case 1 | Case 2 | Case 3 | Case 4 |
|---|---|---|---|---|
| *I/O Ports* | 31 (5%) | 51 (9%) | 73 (14%) | 73 (14%) |
| *Slices* | 199 (1%) | 386 (2%) | 1148 (6%) | 1413 (8%) |
| *DSP48As* | 4 (4%) | 8 (9%) | 16 (19%) | 23 (27%) |
| *RAM Blocks* | 1 (1%) | 4 (4%) | 7 (8%) | 10 (11%) |
| *Max. Freq. (MHz)* | 66.251 | 53.225 | 67.150 | 50.940 |

uses X-coordinate and vehicle orientation as inputs to perform parking maneuvers using only backward movement [36]. Cases 2 and 3 are the hierarchical fuzzy controllers proposed in [35] and [19], respectively. Finally, Case 4 is the proposal analyzed in this paper. Ten bits for input and output precision are used in all the cases. The new technique improves the results of previously reported works. The first-order Takagi-Sugeno defuzzification method included in *XfuzzyLib* allows the hardware implementation of the control strategy described in [35], whereas the capability of *xfsg* to directly synthesize hierarchical fuzzy modules considerably speeds up the implementation described in [19].

## VI. Conclusions

A codesign strategy for FPGA implementation of fuzzy embedded controllers for industrial applications is presented in this paper. It allows the development of hybrid HW/SW control systems that combine a general purpose processor and specific fuzzy inference modules. The design approach is supported by a parameterized cell library that provides optimized building blocks for a wide variety of fuzzy modules, and an automatic synthesis tool for rapid description and implementation of hierarchical fuzzy systems. Combining these elements with a model-based design flow supported by different CAD tools running in the Matlab environment eases the concurrent synthesis and verification of hardware and software components in every design stage. The advantages of the proposed technique are demonstrated through the development of a fuzzy controller for automatic parking of an autonomous vehicle. The described methodology allows accelerating the design process of complex fuzzy controllers. This reduction in development time can be exploited to optimize a particular solution, as well as to compare different solutions in order to obtain the best tradeoff between cost and performance.

## 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). The authors would like to thank the Robotics, Vision, and Control Group from the Engineering School of University of Seville for collaboration to obtain the results with Romeo-4R.

## References

[1] L. A. Zadeh, *Roles of soft computing and fuzzy logic in the conception, design and deployment of information/intelligent systems*, in: O. Kaynak et al., editors, Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, Springer Verlag, 1998.

[2] N. K. Sinha and M. M. Gupta, *Soft Computing and Intelligent Systems: Theory and Applications*, Academic Press, 2000.

[3] A. Abraham, *Intelligent systems: Architectures and perspectives*, in: A. Abraham, L. Jain, and J. Kacprzyk, editors, Recent Advances in Intelligent Paradigms and Applications, Springer Verlag, 2002.

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

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

[6] 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.

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

[8] 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.

[9] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in Industrial Control Applications", *IEEE Trans. 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 Conference on Neural Networks*, July 2001, pp. 189–194.

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

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

[14] 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 Trans. on Industrial Electronics*, vol. 59, no. 2, February 2012, pp. 1208 –1217.

[15] T.-H. S. Li, S.-J. Chang, and Y.-X. Chen, "Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot", *IEEE Trans. on Industrial Electronics*, vol. 50, no. 5, October 2003, pp. 867- 880.

[16] S. Islam, Md. Anwarul, Md. Saukat, and M. Othman, "Design and Synthesis of Mobile Robot Controller using Fuzzy", in *Proc. 28th Int. Conference on Software Engineering*, May 2006, pp. 825-829.

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

[18] D. Kim, "An Implementation of fuzzy logic controller on the reconfigurable FPGA system", *IEEE Trans. on Industrial Electronics*, vol. 47, no. 3, June 2000, pp. 703-715.

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

[20] K. Basterretxea, I. del Campo, and J. Echanobe, "A semi-active suspension embedded controller in a FPGA", in *Proc.2010 Int. Symp. on Industrial Embedded Systems*, July 2010, pp. 69 –78.

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

[22] I. del Campo, J. Echanobe, G. Bosque, and J. M. Tarela, "Efficient hardware/software implementation of an adaptive neuro-fuzzy system," *IEEE Trans. on Fuzzy Systems*, vol. 16, no. 3, June 2008. pp. 761-778.

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

[24] Y.-S. Kung, C.-T. Hsu, H.-H. Chou, and T.-W. Tsui, "FPGA-realization of a motion control IC for wafer-handling robot", in *Proc. 8th IEEE Int. Conference on Industrial Informatics*, July 2010, pp. 493 –498.

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

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

[27] *MicroBlaze Processor Reference Guides*, Xilinx, Inc.
[28] L. M. Reyneri and F. Renga, "Speeding-up the design of HW/SW implementations of neuro-fuzzy systems using the codesimulink environment", *Applied Soft Computing*, vol. 4, no. 3, 2004, pp. 227-240.
[29] I. H. Altas and A. M. Sharaf, "A Generalized Direct Approach for Designing Fuzzy Logic Controllers in Matlab/Simulink GUI Environment", *Int. Journal of Information Technology and Intelligent Computing*, vol. 1, no. 4, 2007.
[30] 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 Int. Conference of Soft Computing and Pattern Recognition*, December 2009, pp. 711-715.
[31] 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.
[32] *System Generator for DSP User Guide*, v12.4, Xilinx Inc., 2010.
[33] M. Brox, S. Sánchez-Solano, and L. L. Delgado, "XFVHDL4: A Hardware Synthesis Tool for Fuzzy Systems," in *Proc. 11th Int. Conference on Intelligent Systems Design and Applications*, November 2011, pp. 385-390.
[34] F. Cuesta, F. Gómez-Bravo, and A. Ollero, "Parking maneuvers of industrial-like electrical vehicles with and without trailer", *IEEE Trans. on Industrial Electronics*, vol. 51, no. 2, April 2004, pp. 257-269.
[35] I. Baturone, F. J. Moreno-Velo, S. Sánchez-Solano, and A. Ollero, "Automatic design of fuzzy controllers for car-like autonomous robots". *IEEE Trans. on Fuzzy Systems*, vol. 12, no. 4, August 2004, pp. 447-465.
[36] S. Sánchez-Solano, E. del Toro, M. Brox, I. Baturone, and A. Barriga, "A Design Environment for Synthesis of Embedded Fuzzy Controllers on FPGAs", in *Proc. IEEE Int. Conference on Fuzzy Systems*, July 2010, pp. 1-8.

**Santiago Sánchez-Solano** received the B.Sc. degree (with honors) in physics 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 with the Computer Center, University of Seville, he joined the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, where he is currently a Scientific Researcher. He is co-author of 2 books and 150 scientific papers, and has participated in 25 research projects funded by different organisms, acting in 7 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.

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

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

He had 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 at the Microelectronics Research Center (CIME-CUJAE), Cuba. His research interests include embedded computing, hardware/software codesign and algorithm acceleration.

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

Since 2002, she has been with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC) or at the University of Seville. Currently, she is a Postdoctoral researcher under '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.

**Iluminada Baturone** received the B.Sc. 5-year degree (with honors) in physics and the Ph.D. degree (with honors) in physics from the University of Seville, Seville, Spain, in 1991 and 1996, respectively.

Since 1990, she has been with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Seville, and since 1992, she has been teaching at University of Seville, where she is currently Associate Professor. She is co-author of 2 books and more than 100 scientific papers, and has collaborated in more than 25 research projects (national and international). Her current research interests include hardware/software co-design, neuro- and fuzzy systems, piecewise-affine controllers, and microelectronic design of crypto-biometric systems.