A Design Environment for Synthesis of Embedded Fuzzy Controllers on FPGAs

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

Abstract— **This paper presents a design environment for the synthesis of embedded fuzzy controllers on FPGAs. It provides a novel implementation technique that allows accelerating the exploration of the design space of fuzzy control modules, as well as a codesign flow that eases their integration into complex control systems and the joint development of hardware and software components. The set of CAD tools supporting this environment includes specific fuzzy logic design tools provided by *Xfuzzy*, FPGA synthesis and implementation tools from Xilinx, and modeling and simulation facilities from Matlab. As demonstrated by the analyzed design examples, the described development strategy takes advantage of flexibility and ease of configuration offered by the different tools to dramatically speed up the stages of description, synthesis, and functional verification of embedded fuzzy control systems.**

I. Introduction

THE capability of fuzzy logic based inference techniques to describe complex system by means of linguistic rules, avoiding the need for mathematical models and providing good results in terms of adaptability and robustness, has motivated and increasing interest for the use of these techniques in the last years and their application to a great number of problems related to automation, industrial control, and decision-making processes [1]–[3].

As a consequence, different approaches to build fuzzy systems, ranging from software implementation using computer programs, to microelectronic realization resorting to application specific integrated circuits or programmable devices, have been proposed [4]. However, even though fuzzy logic can be considered nowadays as a mature technology, the number of design environments and CAD tools able to translate the high-level description of a fuzzy system into an efficient hardware implementation is still very small.

Additionally, the level of complexity reached by many of the current applications of control systems, as well as their requirements of speed, size, and/or power consumption, makes fuzzy inference modules to be conceived as components of a whole system, which (in general) needs to include different hardware and software elements to carry out its function. An appropriated strategy for implementation of embedded fuzzy controllers is to combine a general purpose processor, to cope with conventional processing tasks, system configuration and initialization, and control objective definition, with dedicated hardware to implement the tasks associated with fuzzy inference [5]. The design of this kind of hybrid HW/SW systems requires the development of new methodologies and design tools in order to cut down the design cycle of new products and make them more competitive in market terms.

This paper presents a design environment for the synthesis of embedded fuzzy controllers on FPGAs. The proposed design flow combines fuzzy logic design tools from the *Xfuzzy* environment, FPGA synthesis and implementation tools from Xilinx, and modeling and simulation tools from Matlab. The advantage of the proposed technique is twofold. Firstly, it allows accelerating the exploration of the design space of fuzzy inference modules to be included in control systems. In addition, it facilitates their integration as peripheral of a general purpose 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. Section II presents a novel technique for implementation of fuzzy systems on FPGAs. It is based on the use of DSP development tools from Xilinx in combination with a specific cell library for efficient implementation of fuzzy inference systems. A HW/SW codesign flow for embedded fuzzy controllers and the supporting tools are described in Section III. Implementation results associated to different design examples are compared in Section IV. Finally, Section V summarizes the main conclusions.

II. FPGA Implementation of Fuzzy Controllers

The continuous evolution of programmable logic devices has promoted the use of FPGAs both as prototyping and final platforms of many types of systems, including fuzzy control systems [6]–[12]. Current FPGA families provide a high number of logic resources combined with specific components (high-speed I/O channels, multiport memories, multipliers, processors, etc.). FPGA manufacturers also

This work was partially supported by projects FP7-INFSO-ICT-248858 (www.mobydic-project.eu) from the European Community, TEC2008-04920 from the Spanish Ministry of Science and Innovation, and P08-TIC-03674 from the Andalusian regional Government.

S. Sánchez-Solano is with the Microelectronics Institute of Seville (CNM-CSIC), c/ Américo Vespucio s/n, 41092, Seville, Spain (phone: +34-954466666; fax: +34-954466600; email: santiago@imse-cnm.csic.es)

E. del Toro is with the CUJAE. Ciudad de La Habana, Cuba (e-mail: ernesto@electrica.cujae.edu.cu).

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

I. Baturone and A. Barriga are with the Microelectronics Institute of Seville (CNM-CSIC) and with the University of Seville, Seville, Spain (e-mail: {lumi,barriga}@imse-cnm.csic.es).

provide powerful CAD tools, which speed up the processes of description, verification, and automatic synthesis of new systems, thus allowing the assembly of a complex embedded system on a programmable device (SoPC). Next subsections describe building blocks, tools, and the design flow of a new technique that takes advantage of these advances to simplify the development cycle of fuzzy control modules on FPGAs.

A. Design of Fuzzy Modules with DSP Tools

In the last years, a set of design environments have been developed to ease the construction of digital signal processing (DSP) algorithms on FPGAs. One of those is the System Generator tool (SysGen) from Xilinx [13]. SysGen is based on Simulink, the interactive tool for modeling, analysis, and simulation of dynamic systems integrated in Matlab. It includes a Simulink library, named "Xilinx Blockset", which provides basic building blocks for digital systems, as well as the software required to perform the synthesis and implementation of the algorithms described and simulated in Matlab on Xilinx's FPGAs.

The usual design flow of this tool starts with the schematic description of the system by means of a Simulink model. The verification stage is carried out by simulation, using the signal generation and visualization facilities offered by Matlab. Once the design functionality has been confirmed, SysGen eases the system implementation by translating the model to different kinds of netlist, by generating a bitstream file for the FPGA, or by building the appropriated interfaces to co-simulate the hardware implementation of the controller in combination with a mathematical model of the plant under control.

Using this tool and the building blocks provided by "Xilinx Blockset", a new library called *XfuzzyLib* has been generated. It includes different alternatives to implement each of the basic elements of the active-rule based architecture for efficient implementation of fuzzy controllers described in [12] (Fig. 1). Fig. 2 shows the Simulink library browser utility illustrating the new fuzzy components grouped by functionalities: membership function generators (MFCs), active rule selectors, antecedent connectives, rule memories, defuzzifiers, and control elements.

Modules in "Xilinx Blockset" library admit a set of parameters to define specific aspects of their functionality, like the bus size or 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 that subsystem is instanced in a higher hierarchical level, parameters can be assigned using numerical values or by means of variables. Numerical values of these variables can be later defined using Matlab's command window or an ".m" file. The design functionality can be verified at any time using the simulation facilities from Simulink and its various components to generate excitation signals and capture and display output data.

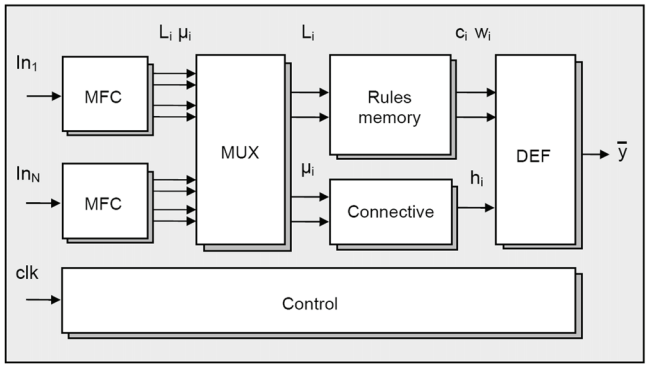


Fig.1. Active-rule based architecture for efficient implementation of fuzzy controllers [12].

Building a fuzzy system with *XfuzzyLib* only requires choosing, interconnecting, and defining the parameters of the needed blocks. When a new system has been defined according to the proposed architecture, the above described design flow can be used to implement it on an FPGA.

In addition to the basic building blocks, *XfuzzyLib* also includes elements describing simple fuzzy logic controllers (FLCs) that differ in the number of inputs, the connective used to calculate the rule activation degrees, and the defuzzification method employed. When a user needs to develop a fuzzy system tailored to a specific application, he/she can employ these simple FLCs or define a new architecture by interconnecting basic building blocks. Also it is possible to hierarchically combine simple FLCs to define more complex fuzzy systems.

Fig. 3-a illustrates some of the FLCs currently available in *XfuzzyLib*, whereas Fig. 3-b shows the block diagram of a 2-

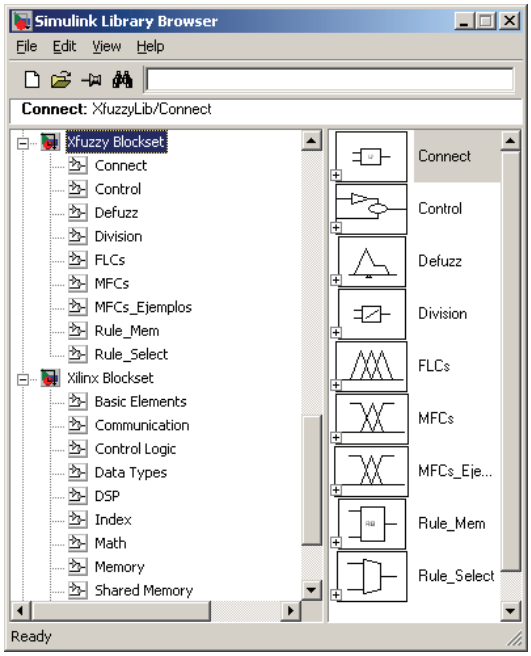
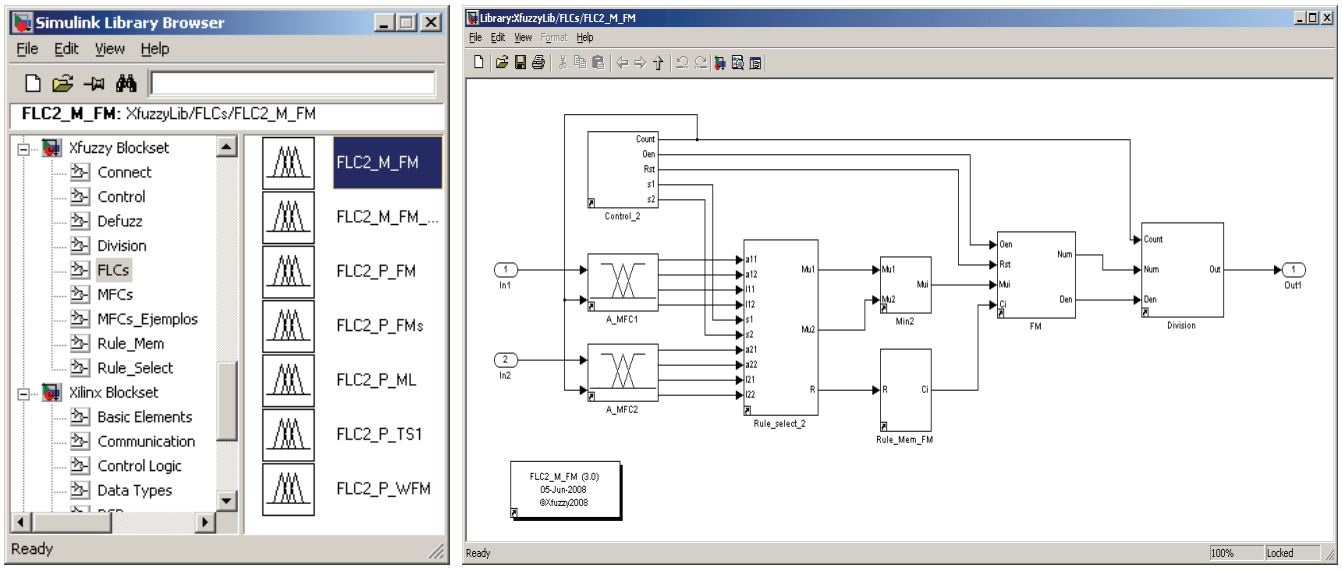


Fig. 2. Components of *XfuzzyLib* integrated in the Simulink Library Browser.



(a) (b)

Fig. 3. a) Fuzzy Logic Controllers (FLCs) included in *XfuzzyLib*. b) Simulink model of a 2-imput 1-output FLC that uses minimum as connective and FuzzyMean defuzzification method.

input FLC that uses minimum as connective and the FuzzyMean defuzification method. Just like basic building blocks, blocks describing FLC architectures are fully parameterizable, making it possible to adapt its functionality according to the requirement of a particular application by introducing the parameters related to the dimension of the system and the definition of its knowledge database. A new synthesis tool incorporated in the *Xfuzzy* environment and described in the next subsection may help to perform this task. Similarly to other library components, also in the case of FLCs the designer can exploit all the facilities offered by Matlab in order to verify the functionality of the inference system.

B. *Automatic Synthesis of Fuzzy Modules with Xfuzzy*

Xfuzzy is a design environment that eases the different stages in the development of a fuzzy inference system. It provides description, simulation, learning, and simplification tools that allow defining, verifying, and optimizing fuzzy systems. The environment also includes synthesis tools to implement them as software or hardware components [14].

When used to develop an embedded fuzzy controller, the fuzzy inference module is described by means of a (usually hierarchical) specification using the *Xfuzzy Specification Language* (XFL3), which combines fuzzy blocks (to perform control and decision-making tasks) and crisp blocks

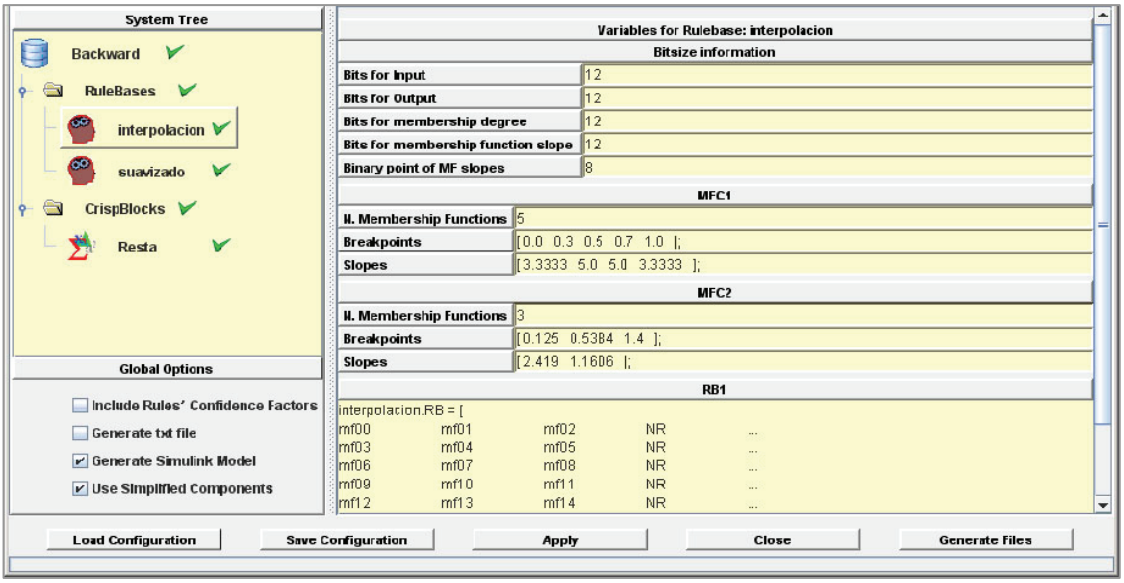


Fig. 4. Graphical user interface of the synthesis tool included in the *Xfuzzy* environment.

(to implement general purpose arithmetic and logic blocks). Knowledge rule bases are directly defined by an expert operator (*xfedit*) or they can be extracted from numerical data using identification (*xfdm*) and/or supervised learning (*xfsl*) tools. Functional verification is carried out by analyzing the input-output relation of the system (*xfplot*), as well as simulating its closed-loop behaviour in combination with a Java codified model of the plant (*xfsim*).

Once validated the XFL3 specification, the *xfsg* synthesis tool, incorporated to the last *Xfuzzy* revisions, is able to generate the files required to start the next design stage. The Graphical User Interface of this tool is shown in Fig. 4. The upper left area contains the knowledge base, structured as a pull-down menu with components grouped under the categories "RuleBases" and "CrispBlocks". When a particular rule base is selected, the right area allows defining the associated parameters and shows information about membership functions and rules directly extracted from the XFL3 specification. When all the fuzzy system components have been configured (identified by green marks close to them) it is possible to generate the output files by pressing the button 'Generate Files'. Basically, *xfsg* generates an ".mdl" file containing the Simulink model of the fuzzy controller, and an ".m" file with the parameters that define the size and functionality of the various components of the fuzzy controller.

C. *Design Example*

The above described design technique 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 instant [15]. 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 park in a predefined position.

Fig. 5-a illustrates the *Xfuzzy* graphical description of a hierarchical fuzzy controller that employs two rules bases and a crisp block to implement a heuristic that allows reverse park the vehicle even when the initial position is near to the objective [16]. The equivalent system description as a Simulink model is shown in Figure 5-b.

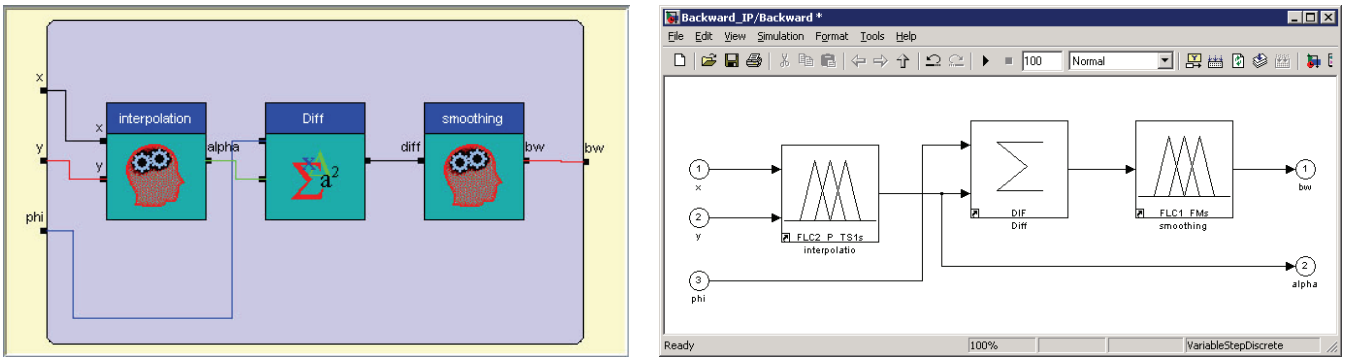
The system functionality can be verified in this stage using the corresponding blocks of Matlab along with its data visualization facilities. Using the "System Generator" block, the fuzzy controller synthesis can be performed. Finally, it is possible to carry out a closed-loop functional verification, combining the controller implementation on an FPGA development board and a software model of the plant, as shown in Fig. 6.

III. Design Flow for Embedded Fuzzy Controllers

As mentioned above, the complexity of current embedded controllers often requires the combined use of general purpose and specific processors. According to the hybrid HW/SW strategy for implementing fuzzy controllers as SoPC proposed in [12], an efficient solution is to use a general purpose processor, MicroBlaze, available as IP-module for the different Xilinx's families of FPGAs, and specific fuzzy inference modules (also developed as IPs) to speed up the inference tasks.

A. *MicroBlaze Processor*

MicroBlaze is a 32-bit RISC processor based on Harvard architecture and available as a soft core optimized for implementation in Xilinx's FPGAs. The system interface for MicroBlaze allows connections to local memory, using the Local Memory Bus (LMB), and to peripheral, by means of the Onchip Peripheral Bus (OPB) and the Processor Local Bus (PLB). There are numerous IP-modules of configurable peripherals compatible with these standards, which allow configuring a specific system according to application requirements. 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 OPB- and PLB-compatible IP-modules [17].



(a) (b)

Fig. 5. a) *Xfuzzy* graphical representation of the hierarchical fuzzy controller used as design example. b) The same fuzzy controller described by means of a Simulink model that uses FLCs and crisp blocks provided by *XfuzzyLib*.

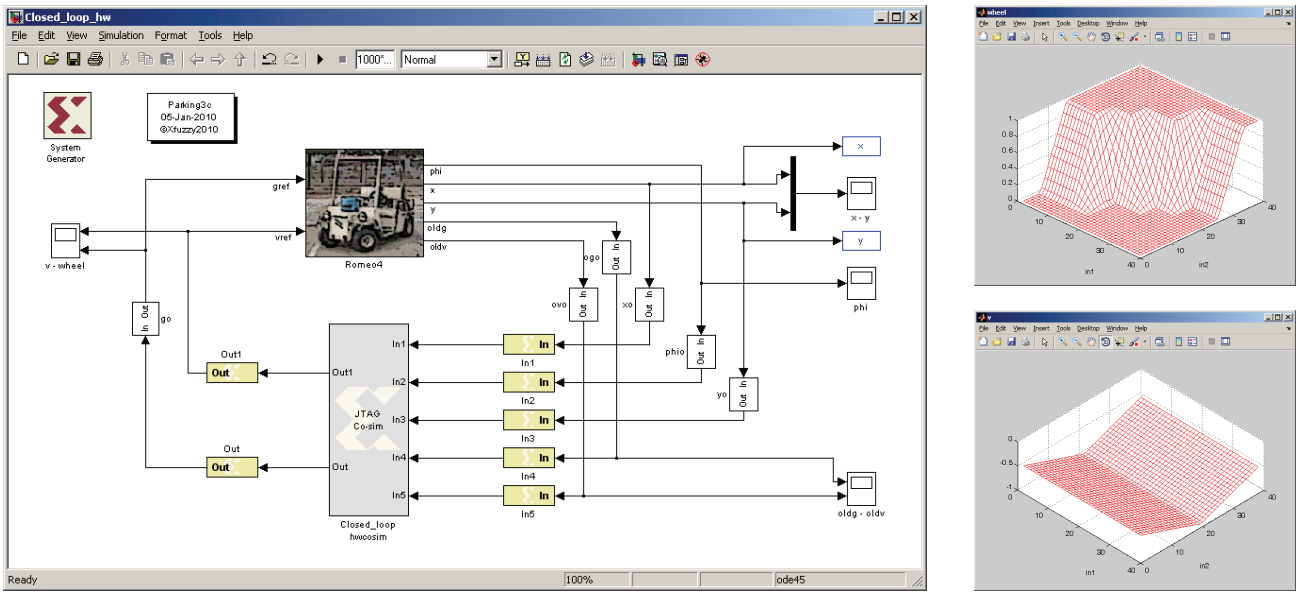


Fig. 6 Simulink model to perform the closed-loop functional verification of Romeo-4R and control surfaces

B. *Design Flow and Tool Chain*

The first stage in the design flow of an embedded fuzzy controller is about the description and functional verification of the fuzzy inference module, as was mentioned above.

Next, assembly of the IP-module and its integration with MicroBlaze processor must be performed. SysGen eases this task as illustrated in Fig. 7. First step is the connection of the input and output signals of the fuzzy controller to shared memory blocks (RAM, FIFO, or registers) for communicating to MicroBlaze. This memory will be mapped onto MicroBlaze address space. Using SysGen's hardware co-simulating capabilities with a Simulink model that contains shared memories, it is possible for the logic design and the host PC software to share a common address space on the FPGA. Next, Simulink MicroBlaze processor block, available in Xilinx Blockset library, must be added. This block uses a previously generated project from XPS and includes the above mentioned processor and other peripheral such as timer and UART.

Import process automates connection between FLC and OPB/PLB interfaces and generates basic drivers to be used in software applications. The GUI of XPS can be seen in Fig. 8. The system configuration view and the software application tab are also visible.

According to the designer needs, various forms of hardware synthesis and hardware co-simulation options can be used to complete the last development step. These

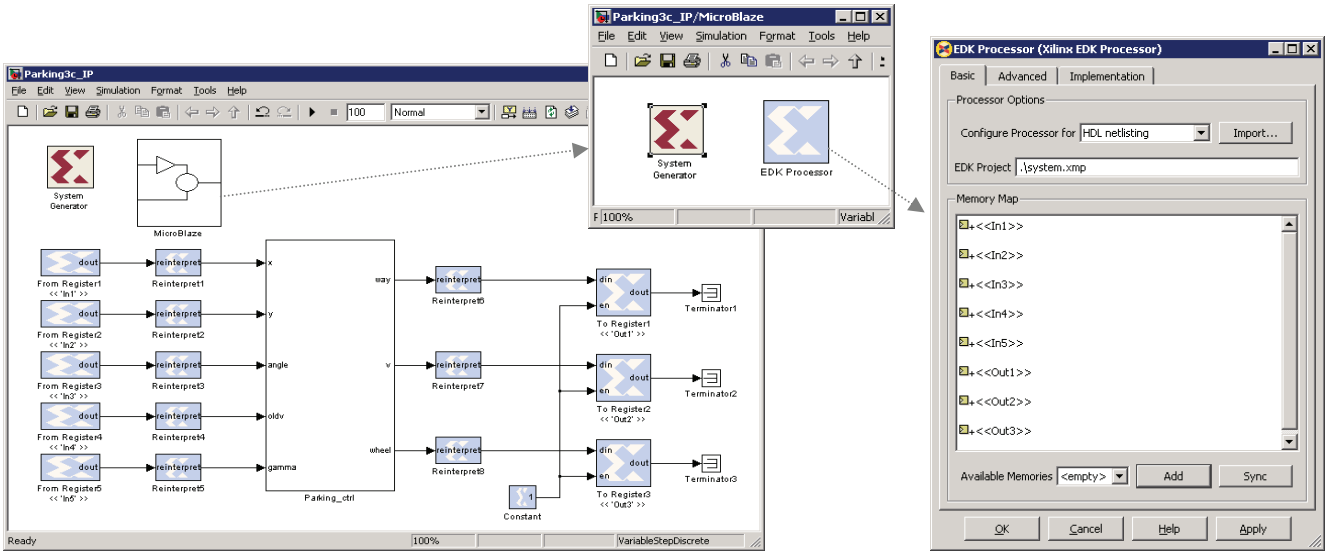


Fig. 7. Integration of the fuzzy inference module with MicroBlaze processor using SysGen.

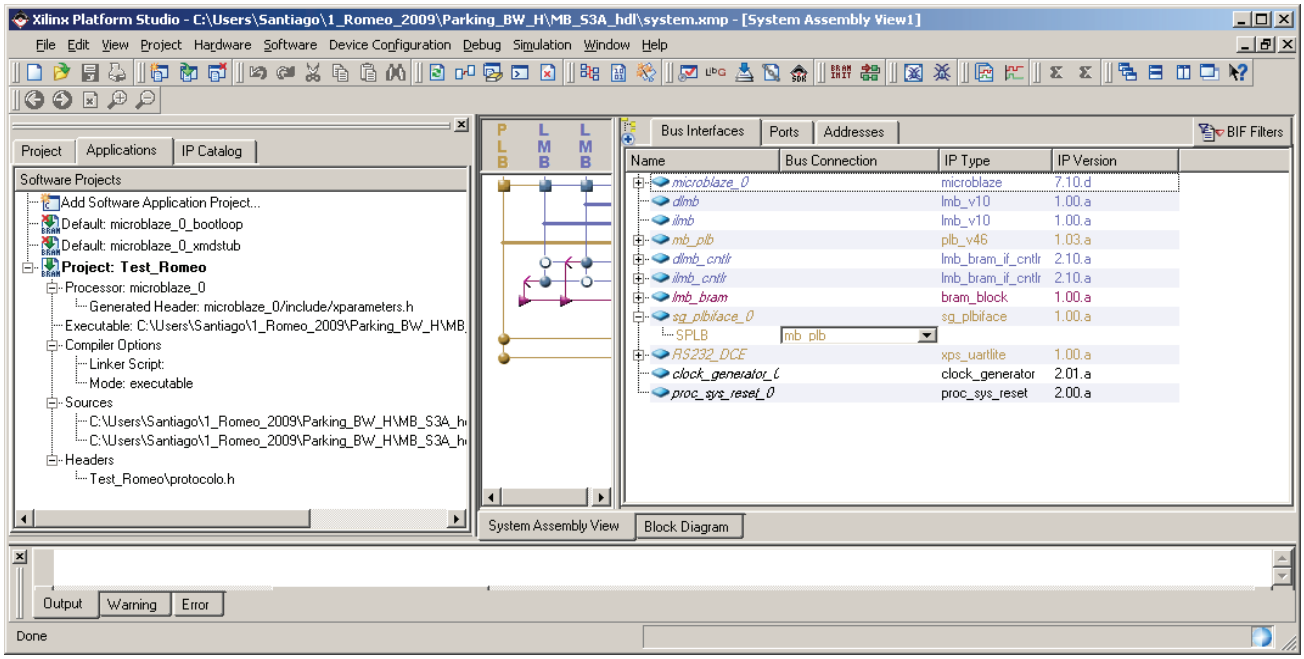


Fig. 8. XPS's graphical user interface showing the System Assembly View and the Applications tab.

options are all available through the "System Generator" block. Using MicroBlaze as a subsystem in a Simulink model (see Fig. 7) enables an FPGA implementation of the processor to be simulated with different fuzzy "peripherals" connected through shared memory blocks in MatLab. This flexibility is of great advantage for verification and tuning of the fuzzy controller.

C. *Design Example*

Romeo-4R uses a digital signal processor (DSP) TMS-320LF from Texas Instruments that provides support for motor control, A/D conversion, and communication links through serial ports, thus easing the low-level control of the vehicle. The DSP acquires information from sensors and processes it by using a kinematical model usually employed for car-like robots in order to estimate the actual position [18].

The fuzzy control module interacts with the DSP using a communication protocol over RS232. The DSP software provides to the fuzzy module with the vehicle status information and, in response, the desired speed and wheel rotation angle is transmitted back as an order to the vehicle.

The joint development of hardware and software components and its connections are illustrated in the Simulink model shown in Fig. 9. "Romeo-4R" block is a model of the vehicle dynamics employed to carry out the system functional verification in a closed control loop. The block tagged "DSP" emulates the TMS-320LF function using a C++ Matlab S-Function implementation. This block includes the communication protocol as well as the corresponding code to communicate to other Simulink blocks in the model. The communication protocol and the software drivers to communicate with the FLC 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. "Parking_ctrl" block provides an interface to test different modules of fuzzy controllers. Finally, some extra blocks are used to initialize parameters, simulation control, and result visualization.

IV. Implementation Results

The proposed implementation strategy and the design flow were successfully applied to implement a control system based on different heuristic strategies. Three fuzzy controllers have been considered. The first one describes a simple functionality that uses only one rule base with X-coordinate and vehicle orientation as inputs, and returns wheel rotation angle as response to perform an effective parking maneuvers using only backward movement. The second choice corresponds to the controller exposed in Section II-C and described in [16]. This system also includes Y-coordinate as input and gives a more effective trajectory. The last case is able to perform back and forth maneuvers in order to include more options in the parking problem. This proposal is better explained in [12]. The controller uses a hierarchical structure containing six rule bases. Three rule bases correspond to decision-making process (in order to differentiate back and forth movement). The other three rule bases correspond to control modules used to calculate the desired speed and wheel angle.

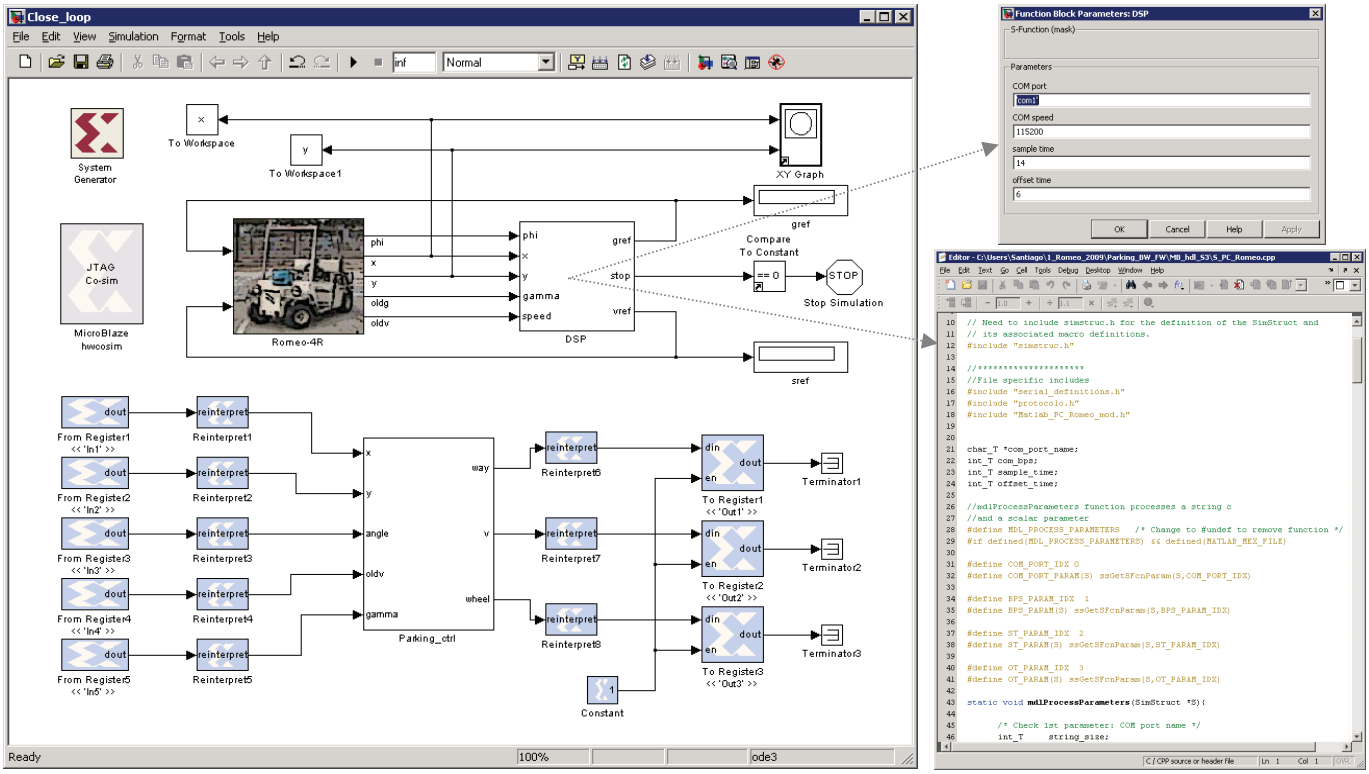


Fig. 9. Joint development of hardware and software components using the proposed design environment.

Results gathered from parking co-simulation with these controllers and *Xfuzzy* results for the same initial conditions are shown in Fig. 10. 'Case 1' corresponds to the simplest fuzzy controller using rules directly extracted from expert knowledge. Rule bases of the controller in 'Case 2' were adjusted by means of tuning tools in order to provide minimal paths in backward trajectories. Finally, the fuzzy controller considered in 'Case 3' allows parking the vehicle combining forward and backward maneuvers.

Table I contains implementation data and FPGA resource utilization in the synthesis of these fuzzy controllers using a Spartan 3A FPGA from Xilinx. Twelve bits for input and output precision were used in all cases (except for output of decision-making blocks in case 3, where only three bits were required to codify the different situations). All the fuzzy control modules are synchronized by the 50 MHz clock available at the FPGA development board.

TABLE I
IMPLEMENTATION RESULTS

	Case 1	Case 2	Case 3
BSCANs	1 (100%)	1 (100%)	1 (100%)
BUFGMUXs	4 (16%)	4 (16%)	4 (16%)
MULT18X18SIOs	4 (20%)	8 (40%)	16 (80%)
RAMB16BWEs	3 (15%)	6 (30%)	9 (45%)
Slices	604 (10%)	873 (14%)	1364 (23%)

V. CONCLUSIONS

A design environment for the synthesis of embedded fuzzy controllers on FPGAs 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 use of a design flow supported by different CAD tool running in the Matlab environment eases the concurrent synthesis and verification of hardware and software components in every stage of design. The advantages of the proposed technique are demonstrated through application development of fuzzy controllers for automatic parking of an autonomous vehicle.

REFERENCES

[1] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 3, n. 1, pp. 28-44. Jan. 1973.

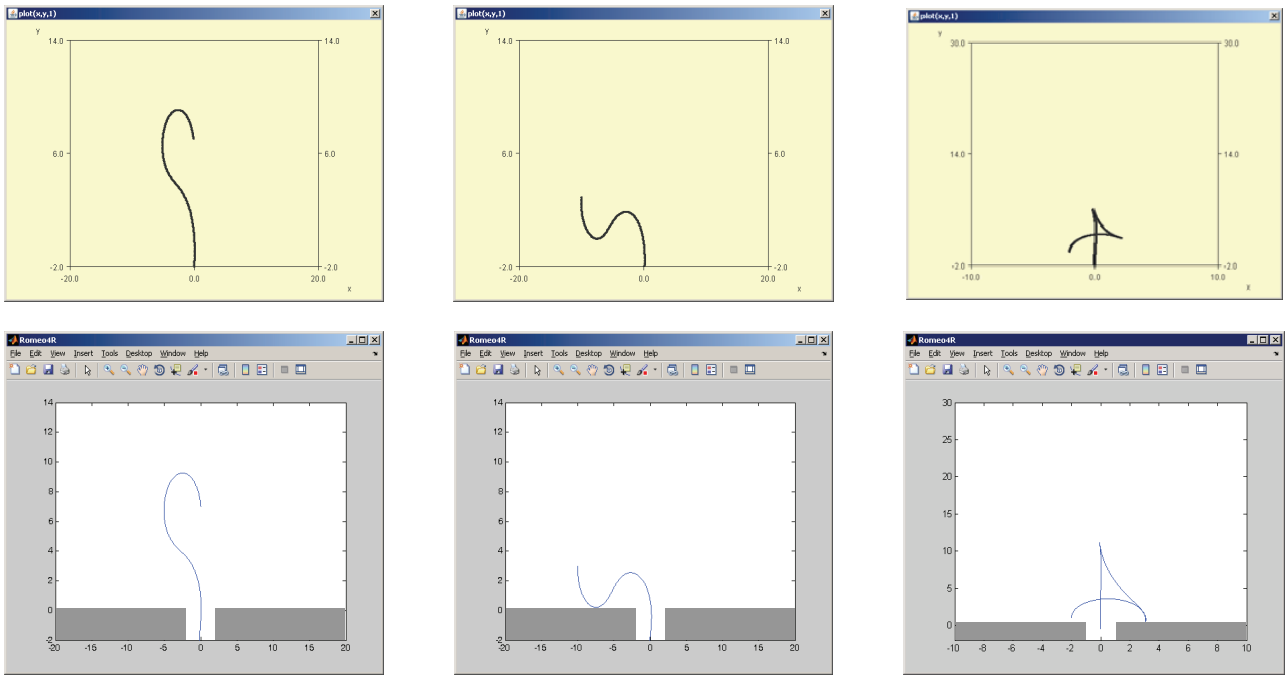
[2] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, 1995.

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

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

[5] 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, n. 3, pp. 271-285, Aug. 2004.

[6] S. Sánchez-Solano, R. Senhadji, A. Cabrera, I. Baturone, C. J. Jiménez, and A. Barriga, "Prototyping of Fuzzy Logic–Based Controllers Using Standard FPGA Development Boards", in *Proc.*



Case 1 Case 2 Case 3

Fig. 10. Parking simulations from *Xfuzzy* (top row) and co-simulation results (bottom row), using three different controllers.

13th IEEE International Workshop on Rapid System Prototyping, pp. 25-32, Darmstadt, Jul. 2002.

[7] D. Kim, "An Implementation of fuzzy logic controller on the reconfigurable FPGA system", *IEEE Trans. on Industrial Electronics*, vol. 47, n. 3, pp. 703-715, Jun. 2000.

[8] 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, n. 5, pp. 867- 880, Oct. 2003.

[9] 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, n. 3, pp. 941-949, Jun. 2006.

[10] J. L. González, L. T. Aguilar, and O. Castillo, "FPGA as a Tool for Implementing non-fixed structure fuzzy logic controllers", in *Proc. IEEE Symp. on Foundations of Computer Intelligence*, pp. 523-530, Honolulu, Apr. 2007.

[11] O. Montiel, Y. Maldonado, R. Sepúlveda, and O. Castillo, "Simple Tuned Fuzzy Controller Embedded into an FPGA", in Proc. Annual Meeting of the North American, Fuzzy Information Processing Society, pp. 1-6, New York, May 2008.

[12] 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, n. 4, pp. 1937-1945. Aug. 2007.

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

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

[15] 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, n. 2, pp. 257 – 269, Apr. 2004.

[16] 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, n. 4, pp. 447-465. Aug. 2004.

[17] *MicroBlaze Reference Guides*, Xilinx, Inc.

[18] 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. On Intelligent Components and Instruments for Control Applications*, pp. 101-106, Jul. 2003.