XFVHDL: A Tool for the Synthesis of Fuzzy Logic Controllers

E. Lago, C. J. Jiménez, D. R. López, S. Sánchez-Solano, A. Barriga

Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica
Avda. Reina Mercedes s/n, (Edif. CICA)
E-41012, Sevilla, Spain

XFVHDL: A Tool for the Synthesis of Fuzzy Logic Controllers

E. Lago, C. J. Jiménez, D. R. López, S. Sánchez-Solano and A. Barriga

Instituto de Microelectrónica de Sevilla. Centro Nacional de Microelectrónica,
Edificio CICA, Avda. Reina Mercedes s/n, 41012-Sevilla, SPAIN.

Abstract

A tool for the synthesis of fuzzy controllers is presented in this paper. This tool takes as input the behavioral specification of a controller and generates its VHDL description according to a target architecture. The VHDL code can be synthesized by means of two implementation methodologies, ASIC and FPGA. The main advantages of using this approach are rapid prototyping, and the use of well-known commercial design environments like Synopsys, Mentor Graphics, or Cadence.

1: Introduction

The two main factors that limit the realization of an electronic system are its complexity and its development time. The use of computer aided design environments including synthesis tools eases the design process and accelerates the introduction in the market of the final product. Concerning the design flow of a fuzzy logic controller (FLC), two different levels may be considered. The algorithmic level specifies the functional behavior of the controller. The objective within this level is to define the shape of the membership functions, the implication mechanism, and the defuzzification strategy that better achieve the proposed control task. At the circuit level, the designer has to select an efficient controller architecture, design the required building blocks, and verify the temporal behavior of the system.

The use of VHDL as a language to support the data structures and functions required for simulation of fuzzy systems has been introduced in different references [1] [2]. In a previous paper, the authors presented a VHDL package which allows formal description and simulation of fuzzy controllers, thus easing the algorithmic design of FLCs [3]. In this occasion our work is focused towards the implementation of FLCs as microelectronic circuits, making use of the capability of VHDL as input language for most of the currently available synthesis tools.

This paper describes a synthesis tool, Xfvhdl, that translates the high level representation of a fuzzy controller in a synthesizable VHDL description suitable for being implemented as a semi-custom application specific integrated circuit (ASIC), or as a field programmable gate array (FPGA). In the second case, the tool also provides script files to drive the synthesis process.

2: Implementation of fuzzy controllers

Since fuzzy logic started to be applied to solve control problems, an increasing number of new applications directed to both industrial and consumer products has emerged [4]. However, the use of fuzzy technologies in real-time control problems demands the development of new processing structures which allow the efficient hardware implementation of inference mechanisms [5].

A low-cost high-speed architecture for fuzzy controllers was proposed by the authors in [6]. The keys for achieving these requirements are the adoption of some restrictions in the shape of membership functions, the use of simplified defuzzification methods, and the use of an active-rule driven inference mechanism.

The block diagram of this architecture is depicted in Fig. 1, showing the three typical components of a fuzzy inference system. Membership function circuits (MFCs) at the fuzzifier stage calculate the degrees of membership for the controller inputs to the fuzzy sets that represent the antecedents of the control rules. Each MFC provides as many pairs "label-activation degree" as overlapping degree has been fixed for the system. The inference stage is composed of an active-rule selection circuit (a counter-controlled multiplexer array is used for this purpose), a multi-input Min circuit that evaluates the rule activation degree by combining the antecedent activation degrees provided by the MFCs, and a rulebase that stores the parameters which define the rule consequents.

There are different options when designing each of the building blocks, with regard to the physical implementation of fuzzy controllers based in the herein described
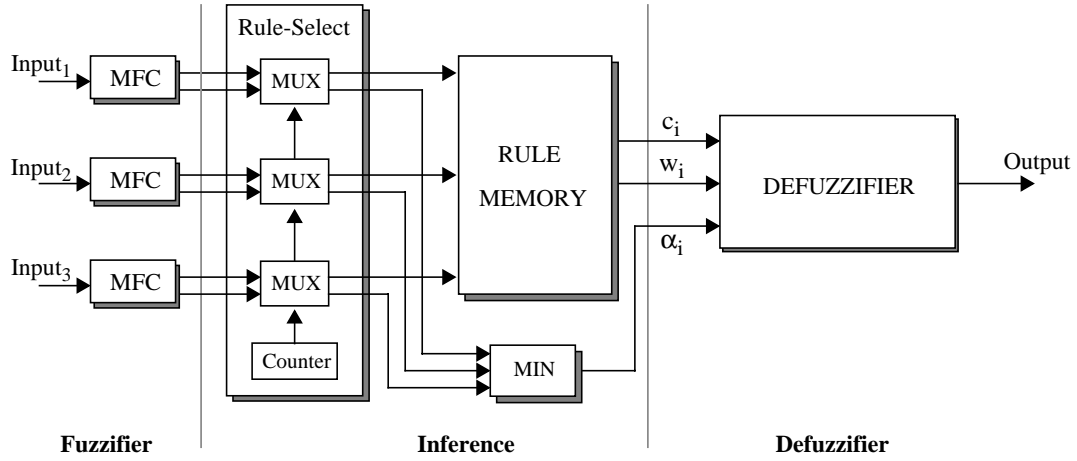
Fig. 1: Active-rule architecture for fuzzy controllers.

architecture. MFCs can be implemented resorting to either a vectorial or arithmetic approach. Memory based MFCs (Fig. 2a) allow a definition of unrestricted membership shapes, but arithmetic approaches (Fig. 2b) provide, in general, better results in terms of silicon area. The rule memory at the inference stage can be implemented by a RAM, in order to improve the controller programmability, or by means of a ROM or a combinational circuit to reduce the area consumption. Lastly, different defuzzification strategies can be considered in the final stage of the controller [7]. The hardware realization of most simplified defuzzification methods (Fuzzy Mean, Weighted Fuzzy Mean, Yager, Center of Sums) requires one of the four structures depicted in Fig 3. Note that sections enclosed by dotted lines are common to all the structures, enabling the construction of a multifunctional defuzzification circuit [6].

The choice among these design options depends on the application domain of the fuzzy controller and the implementation technique used to build the integrated circuit. RAM implementations of MFCs and the rulebase may be a good alternative for a general purpose programmable fuzzy controller. Conversely, the use of ROM or combinational blocks to store the knowledge base is better suited when the controller specifications are well established, or when the programmability is directly achieved by the implementation technique (as in the case of FPGAs). On the other hand, the number of circuits can affect the technology to be employed. FPGAs are a good solution for fast prototyping, while the cost of a high-volume production should suggest the use of semi-custom ASICs.

In order to accelerate the design process, it is helpful to have a design environment which allows the exploration of the design space and eases the automatic synthesis and verification of fuzzy hardware.
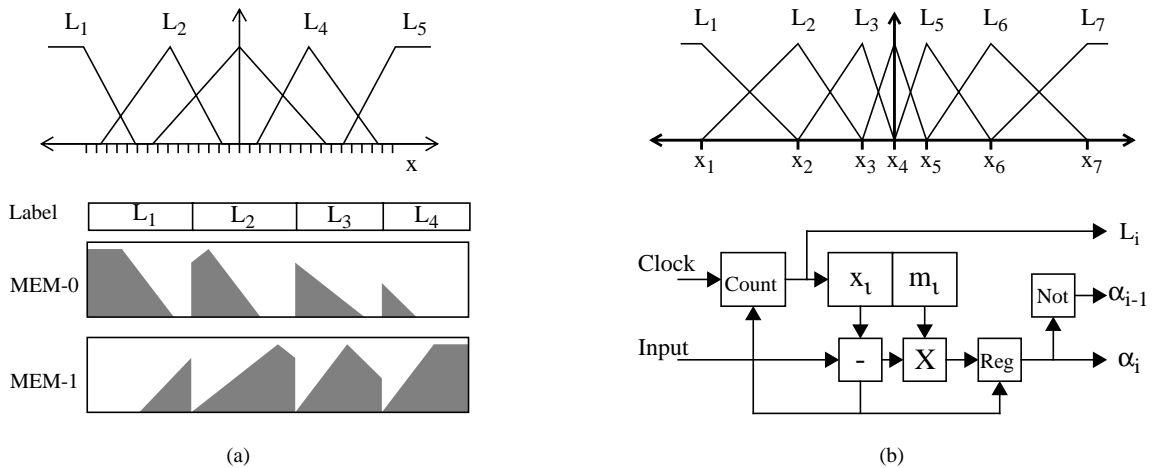

Fig. 2: Implementation of MFCs by vectorial (a), and arithmetic (b) approaches.

$$(a) \quad \hat{y} = \sum_{i=1}^{r} \alpha_i \cdot c_i \Big/ \sum_{i=1}^{r} \alpha_i$$

$$(c) \quad \hat{y} = \sum_{i=1}^{r} \alpha_i \cdot (p_i + q_i \cdot \alpha_i) \Big/ \sum_{i=1}^{r} \alpha_i$$

$$(b) \quad \hat{y} = \sum_{i=1}^{r} \alpha_i \cdot w_i \cdot c_i \Big/ \sum_{i=1}^{r} \alpha_i \cdot w_i$$

$$(d) \quad \hat{y} = \sum_{i=1}^{r} c_i \cdot \left(1 - \beta_i^2\right) \Big/ \sum_{i=1}^{r} \left(1 - \beta_i^2\right)$$
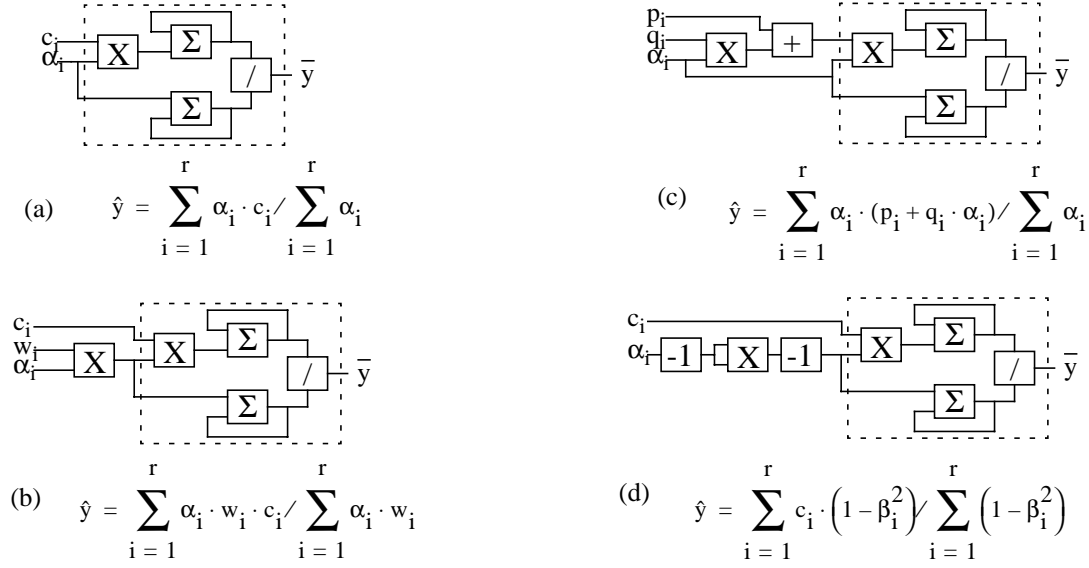
Fig. 3: Block diagrams for different defuzzification methods: a) Fuzzy Mean. b) Weighted Fuzzy Mean. c) Yager. d) Center of Sums with min implication. Where α_i is the activation degree, c_i is the crisp consequent, w_i is a weight parameter, p_i and q_i are shaping factors, and $\beta_i = 1 - \alpha_i$.

3: The Xfvhdl tool

Xfvhdl generates a synthesizable VHDL description of a fuzzy controller from its high-level representation in the XFL language [8]. The XFL specification (Fig. 4) includes information about the behavior of the FLC (knowledge base, inference mechanism and defuzzification method).

The design flow of FLCs using Xfvhdl is illustrated in Fig 5. Xfvhdl uses a cell library containing the parameterized VHDL description for the basic building blocks. There are two kind of blocks: data path building blocks (implementing the inference algorithm) and control blocks (controlling the memory write/read operations, and the signals that control the operation scheduling). The code used in the description of the cell library is compatible with the restricted VHDL implementations of Synopsys and Mentor Graphics tools. The architectural options and the number of bits of precision are defined by the user when the Xfvhdl command is run.

Xfvhdl produces as output the following files describing the FLC:

```
#and min                            system
#composition max                    (Terror ? error, Tdeltaerr ? deltaerr,Toutput ! output)
#defuzzification FuzzyMean
                                     rulebase
type Terror : real [64] (-1<1) {    {
 NN triangle (-1.5,-1,0)               if (error is NN & deltaerr is NN) -> output is ZZ
 ZZ triangle (-1,0,1)                  if (error is NN & deltaerr is ZZ) -> output is NP
 PP triangle (0,1,1.5)}                if (error is NN & deltaerr is PP) -> output is NG
                                       if (error is ZZ & deltaerr is NN) -> output is PP
type Tdeltaerr : real [64] (-1<1) {    if (error is ZZ & deltaerr is ZZ) -> output is ZZ
 NN triangle (-1.5,-1,0)               if (error is ZZ & deltaerr is PP) -> output is NP
 ZZ triangle (-1,0,1)                  if (error is PP & deltaerr is NN) -> output is PG
 PP triangle (0,1,1.5)}                if (error is PP & deltaerr is ZZ) -> output is PP
                                       if (error is PP & deltaerr is PP) -> output is ZZ
type Toutput : real [64] (0<1) {    }
 NG triangle (-0.5,0,0.25)
 NP triangle (0,0.25,0.5)
 ZZ triangle (0.25,0.5,0.75)
 PP triangle (0.5,0.75,1)
 PG triangle (0.75,1,1.5)}
```

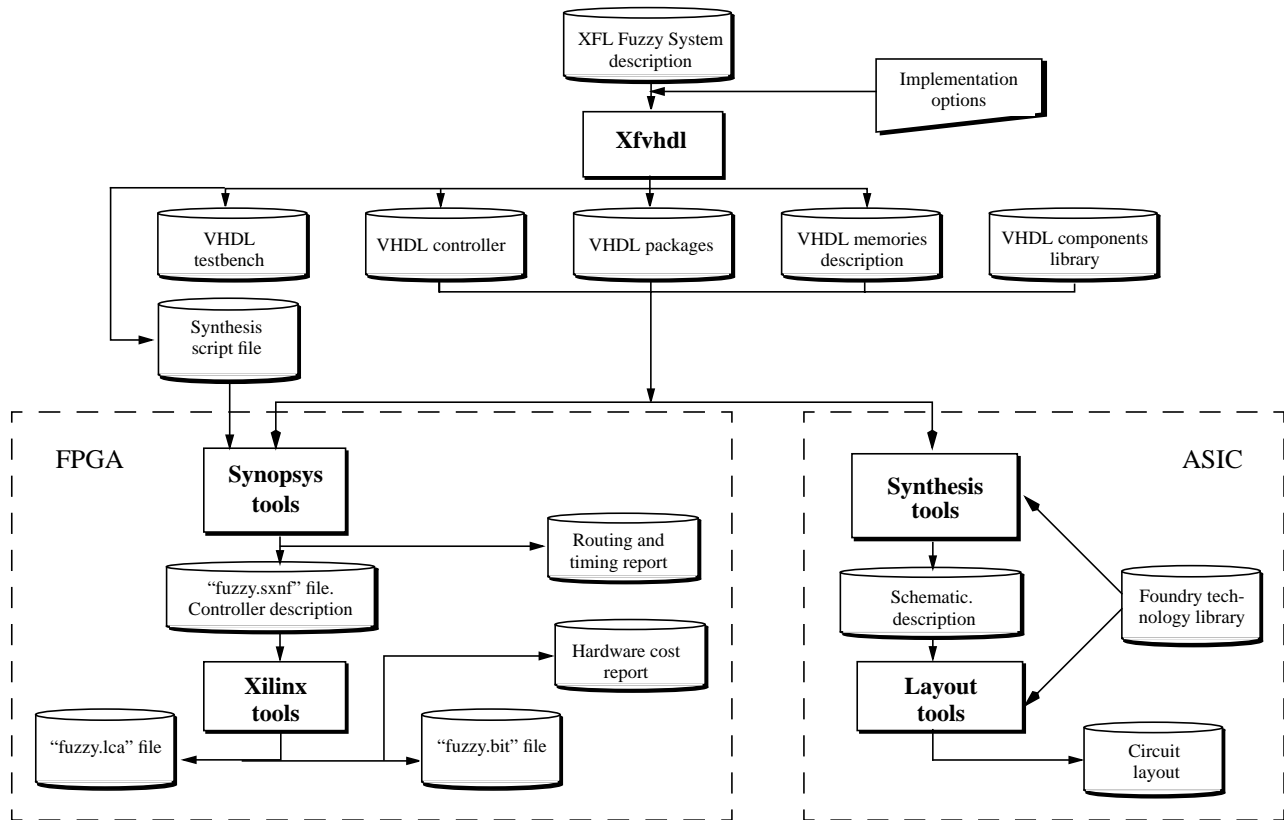Fig. 4: XFL description of a fuzzy controller.

Fig. 5: Design flow for the implementation of FLCs using Xfvhdl.

Package files: Two VHDL packages are generated by Xfvhdl. The *constants* file includes the declaration of the constants used in the VHDL description. Some of these constants are obtained directly from the parameters of the Xfvhdl command. Others are obtained by analyzing the XFL description. The last set of parameters is calculated from the other two. On the other hand, the *entities* file contains the declaration of all the blocks that make up the FLC. The instantiation of each block depends on the architectural options selected.

Knowledge base files: The information about antecedents, rules and consequents is codified in a set of files. The definitions are based in tables of values by means of VHDL "case" sentences, thus enabling logic minimization when these blocks are implemented as combinational logic.

Controller file: This file contains the structural VHDL description of the FLC. The controller is constructed by concatenating a set of basic building blocks according to the XFL description and the implementation options.

TestBench file: In addition to the files required by the synthesis process, a testbench file is generated to ease the verification of the FLC. The testbench includes the instantiation of the FLC, a process to generate a periodical clock signal, and another process that provides the initial reset signal and the inputs used in the simulation of the FLC.

The files generated by Xfvhdl can be used as the starting point for automatic synthesis tools which provide different implementation techniques for integrated circuits. One of the primary decisions in the design process is the selection of the target implementation style. This selection depends on specifications or is based on economical requirements. Later design steps are strongly affected by this choice. Fig. 5 shows the two implementation methodologies marked by dotted lines. The next section describes the realization of several prototypes of ASICs and FPGAs, respectively. In the last case, a *script* file provided by Xfvhdl can drive the synthesis when Synopsys is the selected tool and Xilinx the objective technology.

Xfvhdl can be executed interactively or from the Xfuzzy graphical environment [8]. Fig 6 shows the command line and some Xfvhdl window with its command options. There are more options, not shown, to guide the Synopsys synthesis step.
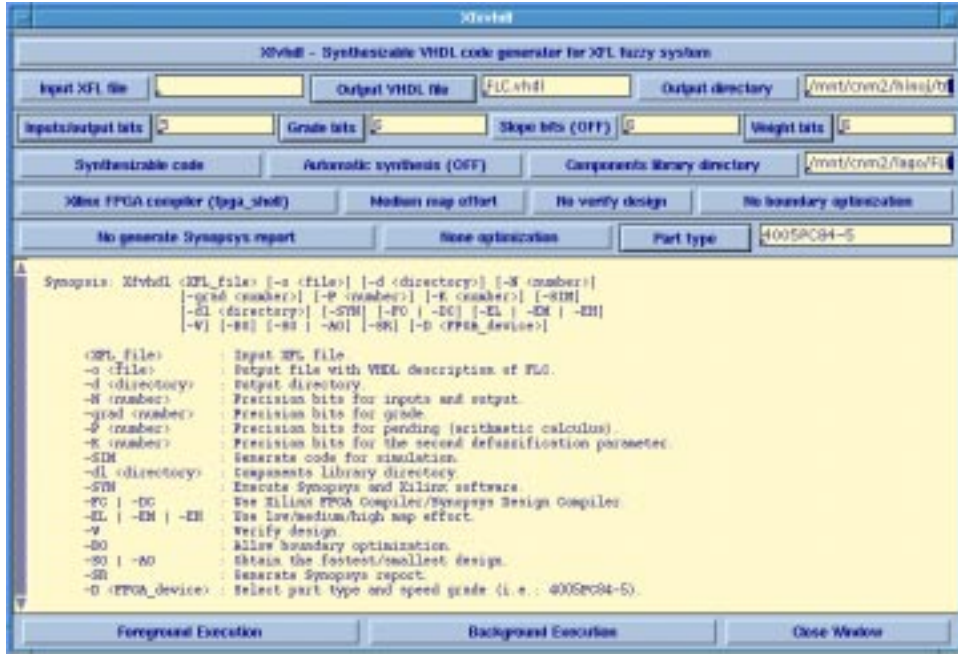
Fig. 6: Xfvhdl graphical user interface showing its main command options.

4: Implementation examples

The ASIC design approach for the architectural component modelling is based on technology mapping criteria. In this sense, FLC implementation requires standard cell and macrocell selection. The VHDL files provided by Xfvhdl are used by the synthesis tool to generate the schematic description of the controller according to the selected foundry library. This schematic is used by place and routing tools to obtain the circuit layout.

An important aspect in the design of ASICs is to adjust the circuit to the required performance under cost and timing constraints. To be able to do this, the designer needs an efficient tool to explore the design space. This is one of the main features of Xfvhdl since its starting point is a high-level behavioral description.

Fig. 7 shows four ASIC implementation examples of a fuzzy controller with varying design parameters. The controller requirements are: two inputs and one output, 7 membership functions, and Fuzzy Mean defuzzification method (eq. (a) of Fig. 3). The figure shows the layout and silicon area of four circuits (implemented in a 0.7 μm CMOS technology) based on arithmetic or memory MFCs, with a different number of bits for inputs and membership grade coding.

Xfhvdl allows for another alternative implementation for fuzzy controllers, based on Xilinx FPGAs. In order to control the design steps, Xfvhdl generates a *script* file for Synopsys to select the synthesis options. The output of Synopsys is an *XNF* file (*Xilinx Netlist Format*) named '*fuzzy.sxnf*' with the controller description. Optionally it is possible to generate a report file containing routing requirement (number of CLBs and IOBs) and temporal constraints.

Finally, *fuzzy.sxnf* is used as the input file for Xilinx software for mapping and routing the FPGA. Three files are obtained as result: '*fuzzy.lca*', '*fuzzy.bit*' and the report of the implementation (in file '*fuzzy.rpt*'). The last step is to write the FPGA using the file '*fuzzy.bit*', to obtain the physical implementation of the fuzzy system from the behavioral XFL description.

Table 1 shows FPGA implementations of the controllers depicted in Fig. 7. Case D exceeded the 4013PQ160-5 FPGA size. As mentioned above, the output generated by Xfvhdl consists of a schematic VHDL for the controller.

5: Conclusions.

Hardware realizations of fuzzy controllers can be improved by using adequate design environments which speed up the processes of design, synthesis and verification of these controllers. A CAD tool focused in the automatic synthesis of fuzzy controllers has been described in this paper. The tool is based on the standard hardware description language VHDL. The viability of the proposed method is shown through its practical application.
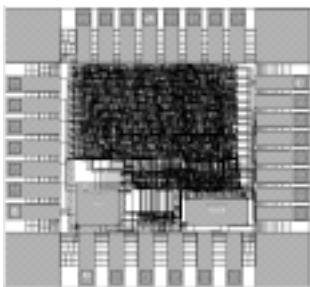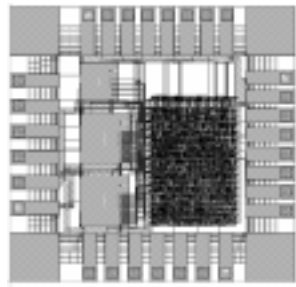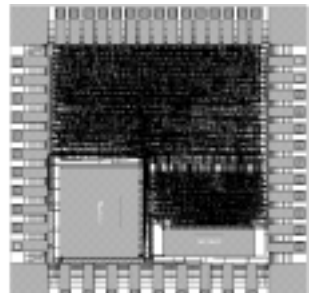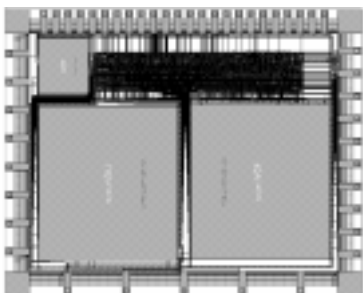
	Arithmetic based MFC	Memory based MFC
4 bits inputs 5 bits membership grade	A) Area = 5.05 mm^2	B) Area = 5.83 mm^2
10 bits inputs 11 bits membership grade	C) Area = 10.96 mm^2	D) Area = 31.66 mm^2

Fig. 7: Examples of the design space evaluation of fuzzy controllers.

	A	B	C
FPGA	4005PC84-5	4005PC84-5	4013PQ160-5
Number of CLBs	162 (82% used)	124 (63% used)	402 (69% used)
Number of IOBs	16 (26% used)	16 (26% used)	34 (26% used)
CLB Flip Flops	116 (29% used)	107 (27% used)	220 (19% used)

Table 1: FPGA implementation for the controllers in Fig 7.

6: References

[1] A. Zamfirescu and C. Ussery, "VHDL and Fuzzy Logic If-Then Rules", *Proc. of Euro-VHDL'92*, pp. 636-641, Hamburg.

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

[3] D. Galán, C.J. Jimenez, A. Barriga and S. Sánchez-Solano: "VHDL Package for Description of Fuzzy Logic Controllers", *EURO-VHDL'95* Brighton, pp. 528-533, Sept. 1995.

[4] T. Munakata, Y. Jani, "Fuzzy Systems: An Overview", *Communications of the ACM*, vol. 37, n. 3, pp. 69-76, Mar. 1994.

[5] D. L. Hung. "Dedicated Digital Fuzzy Hardware". *IEEE Micro*, vol. 15, n. 4, pp. 31-39, Aug. 1995.

[6] C.J. Jiménez, S. Sánchez-Solano and A. Barriga: "Hardware Implementation of a General Purpose Fuzzy Controller". *Proc. 6th International Fuzzy Systems Association World Congress (IFSA'95)*, Sao Paulo, July 1995.

[7] H. Hellendoorn and C. Thomas, "Defuzzification in Fuzzy Controllers", *Journal of Intelligent and Fuzzy Systems*, vol. 1, pp. 109-123, 1993.

[8] D. R. López, S. Sánchez-Solano, A. Barriga: "XFL: a fuzzy logic systems language." *Proc. sixth IEEE International Conference on Fuzzy Systems*, vol. 3, pp. 1585-1591, Barcelona, 1997.