Mathware & Soft Computing 3 (1996) 425-434

Automatic Synthesis of Fuzzy Logic Controllers

A. Barriga, S. Sánchez-Solano, C.J. Jiménez,
D. Gálan and D.R. López
Centro Nacional de Microelectrónica. Dpt. de Diseño Analógico.
Avda. Reina Mercedes s/n Edif. CICA.
41012-Sevilla. Spain.

Abstract

This paper describes a design environment for the hardware realizations of fuzzy controllers which includes a set of CAD tools to ease the description, verification and synthesis of this kind of systems. Special emphasis is focused on the use of a standard hardware description language (VHDL) and compatibility with other integrated circuits design tools.

1 Introduction

Fuzzy inference techniques are becoming an attractive approach to solving control and decision-making problems. This is mainly due to their inherent ability to describe a complex system by means of a simple set of intuitive and ambiguous behavioral rules. The application of fuzzy technologies to real-time control problems demands the development of new processing structures which allow efficient hardware implementations of fuzzy inference mechanisms. In addition, adequate CAD tools are required in order to ease design tasks, and guarantee correct functionality of the final product.

Three different levels may be considered in the design flow of fuzzy logic controllers: algorithmic, architectural and circuit. The algorithmic level specifies the functional behavior of the controller, defining the shape of the membership functions, the implication mechanism, the defuzzification method, etc. Concerning the physical realization, a particular architecture must be selected, and the required building blocks must be identified. Finally, these building blocks will be implemented as an integrated circuit according to the implementation technique more suitable to the specific application. The design tasks at each level are supported by different CAD tools. Traditionally, specific fuzzy tools are used to simulate at the algorithmic level, while architectural and circuit simulations are embedded in conventional circuit design environments.

In this paper we describe a set of CAD tools which provide a unified framework for fuzzy logic controllers design. By means of using a standard hardware description language, this design environment permits verification and automatic synthesis of fuzzy hardware. In section 2, some topics concerning the hardware realization of fuzzy controllers are discussed and the architectural aspects of fuzzy circuits are introduced. Section 3 presents a general overview of the design environment, while the different design levels are described in sections 4 to 6. Finally, an example to illustrate the whole approach is presented in section 7.

2 Hardware Realizations of Fuzzy Systems

In an inference system based on fuzzy logic, the knowledge base is structured as a group of IF-THEN rules:

$$\text{IF } X_1 \text{ is } A_1^r \text{ and } X_2 \text{ is } A_2^r \text{ and... } X_p \text{ is } A_p^r \text{ THEN } Y_k \text{ is } B_k^r \tag{1}$$

where $X_1, \ldots, X_p$ and Y_k represent the system inputs and output, and $A_1^r, ..., A_p^r$ and B_k^r are linguistic labels defined by means of fuzzy sets. These fuzzy sets are characterized by membership functions which may assume different shapes depending on the problem, though in practice, the use of piecewise linear functions is usually sufficient. The controller output is obtained by applying an inference mechanism defined by the connectives used to link the rule antecedents, the implication function chosen, and the rule aggregation operator. The Min operator is usually adopted as the connective for antecedents in fuzzy rules. Concerning implication functions and aggregation operators, the common options are T-norms (Min or Product) and T-conorms (Max or Sum), respectively. A set of well known inference mechanisms are obtained by combining these operators (Min-Max, Dot-Sum, etc.).

The result given by the inference process is a fuzzy set. In control applications, a defuzzifier stage is used to obtain a crisp value characterizing the output fuzzy set. The most commonly used defuzzification methods, Center of Gravity (COG) and Mean of Maxima (MOM) [1], must sweep the whole universe of discourse to provide a solution. From a hardware point of view, this implies the use of massively parallel architectures (which means high area realizations) or sequential techniques (which imply slow system operation).

A considerable reduction in both the inference time and the area of the fuzzy controller can be achieved when using simplified defuzzification methods, where the information provided by the consequents is codified by means of crisp parameters. The overall action of a ruleset is obtained, in this case, by calculating the average of the different conclusions weighted by their grades of activation. Different methods based in this strategy have been proposed in the literature. For the sake of simplicity we will only consider in this paper those methods whose output is given by the general expression:

$$\hat{y} = \sum_{i=1}^{r} \alpha_i \cdot w_i \cdot c_i / \sum_{i=1}^{r} \alpha_i \cdot w_i \tag{2}$$

Several defuzzification methods emerge depending on the choice of w_i in (2). If $w_i = 1$, we obtain the method denominated Fuzzy Mean (FM), its main drawback being that it does not consider the area and support of the output fuzzy set. If

Figure 1: Block diagram for the fuzzy controller architecture.

each w_i represents the area of a consequent fuzzy set the result is the same as in the Center of Sums method (COS). Weighted Fuzzy Mean (WFM) uses weight parameters proportional to consequent supports. On the contrary, the Quality Method (QM) uses parameters inversely proportional to consequent supports to give more importance to crisper, rather than fuzzier, consequents [1].

Defuzzification strategies producing expressions similar to (2) can be implemented using simple arithmetic blocks (two multipliers, a divider, and several adders). In addition, the sumatory in (1) is extended to the number of rules. Taking into account that only the active rules (those with $\alpha_i \neq 0$) will contribute to the solution, the inference time can be drastically reduced if we impose an a priori limitation on the degree of overlapping of the antecedent membership functions.

By adequately combining the above two concepts and introducing pipeline stages an efficient realization of fuzzy controllers can be generated, using the active rule driven architecture shown in Fig. 1 [2]. The membership function circuits (MFC) provide for each input value as many pairs (label, activation level), as the degree of overlapping fixed for the system. Since a fixed degree of overlapping implies restricting the maximum number of active rules, the next step is to sequentially process each of these rules; a counter-controller multiplexer array is used for this. In each counter cycle, the membership degrees are combined through the MIN operator to calculate the activation level of the rule, while the antecedent labels address the memory position containing parameters which define their corresponding consequent. Finally, an arithmetic unit (defuzzifier in the figure) performs the operations of equation (2).

3 Design Enviroment for Fuzzy Controllers

The general scheme of the design environment is shown in Fig. 2. This figure shows also the design flow based on a top-down methodology [3]. The algorithmic description for the controller is made using a formal language called *"Xfuzzy Description Language"* (XFL), as well as through the graphics facilities of Xfuzzy .

Figure 2: General scheme of the tool.

The activities associated to this design stage include choosing the inference mechanism, and defining and tuning the knowledge base (rules, membership functions of antecedents and consequents, etc).

In the next stage the XFL description of the controller is translated into the standard hardware description language VHDL. One advantage of using VHDL is that it is supported by many integrated circuit verification and synthesis tools. The translation process generates the controller circuit structure based on the architecture described in section 2.

After the selection of the architectural parameters and the simulation using the high level data abstraction, a lower level VHDL description is generated. This new description uses the parameterized blocks provided by a cell library. The circuit level VHDL description can be furthermore used as input to (potentially different) tools for automatic hardware synthesis. Depending on the application different integrated circuit implementation techniques such as ASICs, FPGAs, etc. can be applied.

As shown in Fig. 2, each different description level provides simulation stages for verifying the correctness of the design process. At the algorithmic level, a description of the system under control is linked to the controlled specification to validate its behavior. Using Xfuzzy, this description of the system can be done by means of C code, a series of numerical values for the system variables, another fuzzy specification or any combination of these three methods. At the architectural level VHDL itself is used, following the test-bench approach common to the design tools based on the language.

4 Algorithmic Level

Xfuzzy 2.0 is a tool for the development of fuzzy systems. It provides, through a graphical user interface based on X-windows, facilities for the symbolic and graphic specification of fuzzy systems, their set-up by means of simulation and automated learning and a set of different output formats, suitable for both hardware and software implementations.

The different main components of Xfuzzy are shown in Fig. 3a. The core of the system is made up of the XFL library and a simulation shell. Through the services offered by the library, which stores fuzzy specifications in terms of an abstract syntax tree, the definition of the fuzzy system can be manipulated and an implementation suitable for the application domain can be obtained. The simulation shell allows the connection of the fuzzy system under development to one or more modules defining the behavior of the system under control (plant), thus closing the feedback loop.

Fig. 3b shows the set of first-level windows that Xfuzzy offers to the user. The shell is able to integrate data from different sources when performing the controller-plant simulation: C code, numerical series or even a fuzzy specification can be used to model the plant behavior. Each simulation run can be stopped by different end conditions and results can be displayed and/or saved in several (graphical or not) formats. At its current stage, Xfuzzy incorporates a learning module based in backpropagation for the tuning of parameters in membership functions.

During this design phase the user can iterate the cycle, modifying the rulebase or the membership functions to explore the design space. At the end of this phase, the XFL description of the controller stored in the internal representation can be used to generate a functional VHDL description of the system.

5 Architectural Level

The use of VHDL for modeling and simulation is especially appealing since it provides a formal description of the system and allows the use of specific description styles to cover the different abstraction levels employed in the design (architectural, register transfer and logic level) [4].

To extend the capabilities of VHDL towards supporting the description and

Figure 3: Structure and components of Xfuzzy.

simulation of fuzzy logic controllers, the authors have developed a “VHDL package” which includes definitions of data structures that store fuzzy information, and several functions to describe fuzzy inference algorithms [5]. The different elements are defined for integer and real types, allowing the modelling of digital and analog controller realizations.

Basic data structures have been defined to store the values of the controller input and output variables, the membership function descriptions, and the activation degrees of the different rules. Using these types, the package includes another set of more complex definitions which enable the grouping of the data corresponding to the different linguistic variables used by the linguistic rules.

Several functions have also been defined to describe the system database and to carry out the inference mechanism. Several defuzzification methods are provided by means of the *“defuzzifier”* function which provides a single crisp value. Defuzzification strategies considered in this VHDL package include both conventional and simplified methods discussed in section 2.

6 Circuit Level Implementation.

The functional VHDL description of the controller needs to be translated into synthesizable VHDL. In this translation process the user can choose between different alternatives. These alternatives are the antecedent membership function generation mode (either using memory or an arithmetic circuit), the rulebase definition (either using RAM/ROM or a combinational circuit), and the defuzzification method used (currently it is possible to select one among four different methods, or a programmable defuzzifier which implements all of them).

The synthesis process uses a cell library containing the VHDL description for the basic building blocks. These blocks are based on generic parameters chosen by the user. There are two kinds of blocks: data path building blocks (implementing

Figure 4: a) Graphic description. b) Rulebase. c) Membership function definition.

the inference algorithm) and control blocks (controlling the memory write/read operations, and the signals that control operation scheduling).

The implementation techniques used, as well as the different design options to be selected, depend, fundamentally, on the application domain of the fuzzy controller and the number of circuits to be produced. There are a wide range of solutions, some of them are: 1) realization of general purpose systems with programmable membership functions and rulebase using RAM, and the use of programmable defuzzifier; 2) development of FPGA prototypes with a fixed knowledge base and defuzzification method; and 3) implementation of a specific application circuit (ASIC), with the knowledge database in ROM, or generated by a combinational circuit, and a defuzzification method tailored to the application.

7 Example

This section demonstrates the use of the design framework, applying it to a typical example of fuzzy control found in the literature [6]. The purpose is to control the trajectory of a truck as it backs to a loading dock (Fig. 4a). The aim is that the truck arrives at the platform at a 90^o angle and in such a manner that its position locks into the space provided for that purpose. Let us assume that there is sufficient distance in the Y-axis between the initial position of the truck and the destination so as to neglect the Y coordinate, thus the truck movement depends exclusively on its X-coordinate and the angle (PHI) that forms the longitudinal axis of the truck with the horizontal axis. These two variables are the input to the fuzzy controller which, according to the rules shown in Fig. 4b, supplies as output the new direction which the truck wheels should assume (PSI). Fig. 4c shows the membership functions used for antecedents and consequents. This information is obtained by Xfuzzy using a test-and-error method. For more complex problems Xfuzzy provides tools with learning capabilities for membership function tuning.

This example considers a very simple model of the controlled system. Assuming that the truck moves a fixed distance "d" in each inference step, the new coordinates are calculated as a function of the former coordinates and the angle determined by the fuzzy controller output, according to the following equations:

$$\begin{aligned} \mathrm{phi}_t &= \mathrm{phi}_{t-1} + \mathrm{psi} \\ x_t &= x_{t-1} + d * \cos(\mathrm{phi}_t) \\ y_t &= x_{t-1} + d * \sin(\mathrm{phi}_t) \end{aligned}$$

There is a direct mapping from the algorithmic XFL definition of the controller to the VHDL description of its architecture. The VHDL description contains a process which describes the algorithm of the system operation. The specification of the fuzzy logic controller architecture includes the following steps: 1) initialization of membership functions and input and output variables; 2) description of the rulebase and execution of the inference process; and 3) defuzzification and assignment of results to the output.

Figure 5: Simulation results at different levels: a) Xfuzzy; b) VHDL

The graphical results obtained from the Xfuzzy simulation of the system and by the VHDL verification are shown in Fig. 5. While Xfuzzy provides mechanisms for obtaining this graphical output, for the VHDL stage the inclusion of specific sentences in the code modeling the plant is necessary to produce data suitable for the graphical representation of the trajectory. The results shown in Fig. 5 indicate that, for this example, the goal is always reached.

Finally, Fig. 6 shows the layout for a general purpose controller that can be applied to the example discussed here. This circuit uses arithmetic evaluation for the antecedents and contains a programmable defuzzifier. The rulebase and the membership functions for the antecedents are stored in RAM, allowing the controller to be adapted to different applications. The design is based on an 1.0 μm CMOS technology. The total silicon area is 17.40 mm^2.

Figure 6: Fuzzy controller layout.

References

[1] H. Hellendoorn y C. Thomas, "Defuzzification in fuzzy controllers", *Jour. of Intelligent and Fuzzy Systems,* vol. 1, pp. 109-123, 1993.

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

[3] Jiménez, C.J., Galán, D., Barriga, A. y Sánchez-Solano, S.: "Síntesis automática de sistemas de control basados en lògica difusa". Proc. V Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF'95), pp. 263-268, Murcia, Sept. 1995

[4] A. Zamfirescu, "Logic and Arithmetic in Hardware Description Languages", "Fundamentals and Standards in Hardware Description Languages" Edited by Jean P. Mermet, NATO ASI Series, pp. 109-151, 1993.

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

[6] B. Kosko, "Neural Network and Fuzzy Systems", Prentice Hall, 1992.