Abstract

The particular architecture of fuzzy systems has led to the introduction of specific simulators on the market, usually isolated from design environments. This article presents a VHDL package that allows high level descriptions and simulations of fuzzy controllers. The importance of this package to fuzzy hardware design lies not only in its portability to any VHDL simulator, but also in that it allows verification of the simulation results of a particular system in a unique and standard simulation environment, from algorithm description level to RTL level (synthesizable) or logic gates. Finally, an example is included to demonstrate the package functionality.

1: Introduction

Hardware implementation of electronic systems requires the extensive use of verification stages throughout the design process to guarantee correct functionality of the final product. 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 used in the design (architectural, algorithmic, RT and logic level).

As consequence of its adaptability to complex control problems, the interest in fuzzy logic controllers (FLCs) has grown in the last years. As yet, these systems have been simulated using specific CAD tools, independent of the design environment. However, some authors have investigated the possibility of using VHDL as a language to support the data structures and functions required for fuzzy system simulation [1] [2].

This communication presents a VHDL package which allows formal description and simulation of fuzzy controllers. The advantages of this approach derive from first, the use of a standard [3] (enabling portability of the descriptions to any of the market-available VHDL simulators) and second, its integration into VHDL-based design environments (in such a way that simulation results at circuit and algorithmic levels can be compared immediately).

2: Basic concepts of fuzzy logic

The application of approximate reasoning techniques has emerged as a valid alternative in control problems where the system is not easily described by an analytic model [4]. In an inference system based on fuzzy logic, the knowledge base is structured as a group of IF-THEN rules:

\[
\text{IF } X_1 \text{ is } A_1' \text{ and } X_2 \text{ is } A_2' \text{ and... } X_p \text{ is } A_p' \text{ THEN } Y_k \text{ is } B_k'
\]

where \( X_1, ..., X_p \) and \( Y_k \) represent the system inputs and output, and \( A_1', ..., A_p' \) and \( B_k' \) 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 [5] [6].

Figure 1 shows the structure of a fuzzy controller together with the controlled system. An FLC consists of a group of rules and an inference mechanism capable of processing such information. In addition to these basic ele-
ments, two interface blocks are needed to perform the transformation between fuzzy values used in the controller and deterministic values that make up the input and output of the controlled system (fuzzifier and defuzzifier).

The most widely used inference mechanism is that proposed by Mamdani, which uses the maximum and minimum as basic operators [7]. Figure 2 presents an example of FLC operation showing the case of two rules whose antecedents are symmetric triangles. The membership degree of the input to the antecedents is determined in the fuzzification stage. The activation degree of the rule is obtained as the minimum of the membership degrees. The fuzzy set resulting from truncating the consequent of each rule to the value of its activation degree determines the partial conclusion given by this rule. Finally, the rules are aggregated to obtain the controller output. Since the result of the inference process is a fuzzy set, a defuzzification stage must be included to obtain a concrete output value (Y) [8].

3: VHDL package for FLCs.

To extend the capabilities of VHDL to support description and simulation of FLCs, our package includes definitions of data structures that store fuzzy information, and several functions to describe fuzzy inference algorithms. The different elements are defined for integer and real types, that allow modelling digital and analog controller realizations.

3.1: Data structures.

Basic data structures have been defined to store the values of the input and output variables, the membership function descriptions, and the activation degrees of the different rules. Figure 3 presents a schematic of these data structures.

![Figure 3: Basic data types.](image)

The interface between the FLC and the process under control is characterized using the “input_array” and “output_array” types. Both correspond to arrays whose elements contain the input and output values. A membership function is described by the record (“mfc”) formed by five fields: “name”, “data”, “centr”, “area”, and “shape”. The first field stores the linguistic label that identifies the membership function. The second notes its description. The third and fourth fields refer to its centroid and area, and the last one represents its function’s format, which can be one of the following: Triangular, Trapezoidal, Gaussian, or Cauchy. Finally, the activation degrees of the rules are stored in another record-type object named “goa-record”, composed by three fields: “goa”, “consec”, and “num_gra”. The first and the second are arrays where the activation degree and the label of the affected output variable’s membership function are stored, respectively. The number of pairs (goa-consec) located on these two arrays is stored in the last field.

Using these types, the package includes another set of more complex definitions which enable grouping the data corresponding to the different linguistic variables appearing in the rules (Figure 5). The “mfc_set” groups the definitions of the membership functions associated with a variable, while “mfc_all_sets” defines the complete data-
base of membership functions used in the fuzzy controller.

Linguistic variables are defined using “L_var”. This is a record consisting of the following fields: “name” (variable name); “num_mfc” (number of membership functions); “mfcs” (array with the definition of membership functions); “min_value” and “max_value” (minimum and maximum values of the variable’s universe of discourse); “int_step” (value used for the numerical integration in the defuzzification); “discret” (discretization of membership degrees); and “actual_value” (actual value of the variable).

Output variables are described by “Out_L_var” which, in addition to including “L_var”, contains information about the activation degree of the rules, the defuzzification mechanism, and the aggregation and implication functions. It can be described using a record with six fields: “ling_var” (“L_var” type); “GradeOfActiv” (which stores the pairs goa-consequent_label obtained from the rule base inference); “def_method” (where the defuzzification method is chosen); “psi_value” (parameter used in the psi-quality defuzzification method); “agr_method” (where the aggregation function is selected); and “imp_method” (where the implication method is chosen). Implication functions are Minimum, Product, Boundary Product, and Drastic Product. Aggregation functions are Maximum and Sum.

Input variables are jointly associated in a variable “input_var” and the output in “output_var”. In both cases, these are arrays whose elements are “L_var” and “Out_L_var” types, respectively.

3.2: Functions.

Once the available data structures are introduced, we will describe the functions that handle them. First, there are initialization functions that allow describing the system’s database. “init_all_mfc” and “init_all_vars” are functions which use data files to initialize all the membership functions and all the linguistic variables, respectively. “init_mfc”, “init_inputs”, and “init_outputs” are other initialization functions to initialize the membership functions and the linguistic variables with data inside the VHDL code.

The next group of functions implements the inference mechanism. “fzis” performs the fuzzification, that is, it returns the membership degree of the input variable to a linguistic label. “fzand” and “fzor” are the connectives of the different rules. Lastly, “fzthen” performs the fuzzy implication function and the aggregation of the conclusions of the different rules.

Several defuzzification methods are provided by the “defuzzifier” function, which evaluates the controller output according to the values stored in the “def_method” field of the output variables, returning a single, crisp value. Defuzzification strategies considered in this VHDL package include both conventional (Center of Area, Mean of Maximum, etc.) and simplified (Fuzzy Mean, Weighty Fuzzy Mean, Level Grading Method, Yager, etc.) methods [8].

The “defuzzifier_all” function can be used when several output variables are needed. It returns an array-type object with the defuzzification’s crisp values of each output variable, ordered by the initialization sequence.

4: Application to an example.

This section demonstrates the use of the proposed package, applying it to a typical example of fuzzy control found in literature [9]. The purpose is to control the trajectory of a truck as it backs to a loading dock (Figure 6).
The aim is that the truck arrive to the platform at a 90° angle and in such a manner that its position locks into the space provided for such 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 reject the coordinate Y, thus the truck movement depends exclusively on its X-coordinate and the angle (PHI) formed by the longitudinal axis of the truck with the horizontal axis. These two variables are the input of the fuzzy controller which, according to the rules shown in Figure 7, supply as output the new direction which the truck wheels should assume (PSI). Figure 8 shows the membership functions used for antecedents and consequents.

The VHDL description of the FLC is shown in Figure 9. Basically, it contains a process which describes the algorithm of the system operation. The specification of the FLC 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 output.

An “input_var” variable is defined in the declarative part of the FLC process to represent the controller input and an “output_var” to identify the output. In addition, three
“mfc_set” variables are defined to store the descriptions of the different membership functions ("mfc_set_PHI", "mfc_set_X", and "mfc_set_PSI").

The body of the process starts assigning values to the membership functions through the function “init_mfc”. Then the linguistic variables are initialized using the functions “init_inputs” and “init_outputs”, assigning PHI to “vars_IN(1)”, X to “vars_IN(2)”, and PSI to “vars_OUT(1)”.

The descriptions of the control rules and defuzzification are inserted in an endless cycle that includes a “wait” sentence, so that the system will apply the inference mechanism each time any of the input are modified. The specification of each rule uses the “fzis”, “fzand”, and “fzthen” functions. “fzis” calculates the membership degree of the input to the linguistic label passed as argument. “fzand” implements the linking connective between the two antecedents and returns the activation degree of that link. Lastly, “fzthen” applies this activation degree to the linguistic label of the consequent to evaluate the partial conclusion of the rule. Before ending the cycle, defuzzification is performed using the function “defuzzifier” and the resulting value is assigned to the FLC output.

5: Simulation results.

VHDL descriptions of the fuzzy controllers can be validated with any VHDL simulator, using a “test_bench”. To simulate the pair consisting of the controller and system under control, the test_bench must include, in addition to the FLC description, a second VHDL unit which models the controlled system behavior.

This example considers a very simple model of the controlled system that can be described with functions given by VHDL mathematical packages. 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:

\[
\begin{align*}
\phi_t &= \phi_{t-1} + \psi \\
x_t &= x_{t-1} + d \times \cos(\phi_t) \\
y_t &= x_{t-1} + d \times \sin(\phi_t)
\end{align*}
\]

The code describing the system under control includes a series of sentences that send to a file the data needed to obtain a graphic representation of the trajectory followed by the truck for different starting points. Figure 10 shows some of the results obtained using different defuzzification methods. As can be observed in the figures, the target is always reached, regardless of the method employed. However, from a microelectronic point of view, the use of simplified methods eases the hardware realization of fuzzy controller, as indicated by the authors in [10]. In fact, our future work will focus on the automatic synthesis of fuzzy controllers from their VHDL description.

6: Conclusions.

A VHDL package has been presented which permits describing control systems based on fuzzy logic. In addition to the inherent advantages of using standard description language, this approximation allows reducing the gap found between specific simulation tools for fuzzy systems and VHDL-based tools for hardware development. The viability of the proposed method is shown through its practical application.

7: References


Figure 10: Simulation results with different defuzzification methods: a) Center of Area; b) Fuzzy Mean.