XFUZZY: A DESIGN ENVIRONMENT FOR FUZZY SYSTEMS

D. R. López, C. J. Jiménez, I. Baturone, A. Barriga, S. Sánchez-Solano

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

Seventh IEEE International Conference on Fuzzy Systems (FUZZ-IEEE’98),
Abstract

Xfuzzy is a CAD tool that eases the development of fuzzy systems from their conception to their final implementation. It is composed of a set of modules and programs that share a common specification language and cover the different stages of the design process. Modules for describing, verifying and tuning the behavior of the system are integrated within the environment. In addition to these features, common to other fuzzy design tools, a relevant characteristic of Xfuzzy is that it includes several synthesis facilities for implementing the system on either software or hardware.

1. Introduction

The success of fuzzy logic applications in fields such as decision-making systems, control, image recognition and non-linear systems modelling has motivated the introduction of many fuzzy system development tools. However, most of these tools are tightly associated to specific architectures, producing implementations for predetermined microprocessors or microcontrollers, and offering a limited set of fuzzy operations (implication functions, fuzzy connectives, defuzzification methods, etc.). Furthermore, although these tools provide facilities for simulating the behavior of the system under development, they do not have efficient links to conventional integrated circuit design tools, which would be advantageous in the case of hardware implementations.

The fuzzy system design environment presented in this paper intends to solve the above described limitations. To accomplish these goals it uses a specification method which is independent of the particular operations used in each case and which does not impose any limitation on the complexity of the fuzzy system, either on the rulebase structure or on the construction of the rules themselves. Moreover, this specification method is flexible and general enough to be exploited by the different verification and synthesis tools integrated into the design environment. Lastly, the environment has been designed with openness as a main objective, thus allowing the incorporation of new formalisms and methods for system design and verification, as well as the integration of new target (software or hardware) architectures for system implementation.

2. The Xfuzzy Environment

Xfuzzy is a design environment that eases the specification and verification of fuzzy systems, and their final implementation. The modules integrated into Xfuzzy are based on the XFL language [1]. The power and flexibility of this language enable the use of Xfuzzy in a wide range of applications: from the evaluation of different fuzzy operators to the synthesis of fuzzy logic based systems [2].

Figure 1 shows the general structure of Xfuzzy. The kernel of the environment is formed by a set of common func-

![Fig. 1: General structure of Xfuzzy](image)
tions called the XFL library. The elements of this library perform the parsing and semantic analysis of XFL specifications and store them using an abstract syntax tree. This is the common format used inside the environment when handling system descriptions. The modules in charge of the successive design stages lay around the kernel library, using its services. On top of these modules, the environment has a graphical user interface providing a simple and intuitive access to its elements. The user interface is based on X-Win-
dow, using the Athena-3D toolkit. The current version of Xfuzzy runs on any Unix-compatible operating system with X-Window.

Figure 2 shows the general design flow in Xfuzzy. Systems are described in the XFL language by using any text editor or the graphical editors integrated into the environment. An XFL specification contains information about the knowledge base of the system and the mechanisms of inference to be used.

The behavior of the system can be verified using xfsim. This module allows the combination of the system under development with other elements defining its operational context. System definition can be adjusted with xfbpa, a supervised learning module able to tune the parameters of XFL definitions of any complexity [3].

Synthesis facilities constitute the most remarkable feature of Xfuzzy with respect to other fuzzy system development tools. If software synthesis is required, the fuzzy inference engine can be built into a C module by means of xfc. For hardware synthesis, Xfuzzy provides two modules enabling the implementation of fuzzy systems defined in XFL. xftl translates an XFL description into a look-up table that can be implemented on a Field Programmable Gate Array (FPGA). As a second alternative, xfhdl translates an XFL specification into a VHDL description based on a specific architecture. The VHDL code can be further synthesized as an Application Specific Integrated Circuit (ASIC) or as an FPGA.

3. Fuzzy System Description

The description of a fuzzy system in XFL consists essentially of three parts:

- Selection of the fuzzy operations: fuzzy connectives, implication function, aggregation mechanisms and defuzzification method.
- Definition of universes of discourse and membership functions, via XFL types.
- Specification of the system behavior: input/output variables and rulebase, employing XFL modules.

For each of these parts, the Xfuzzy environment offers graphical editors to ease the definition of the system. Fuzzy operations can be selected from a list of those defined in that

4. Fuzzy System Verification

System verification is usually an iterative process that can be performed in Xfuzzy using two different, but not incompatible, strategies: simulating system behavior within its operational context and tuning its parameters using a set of training data.
4.1. Simulating XFL-based Systems

The simulation module `xfsim` allows the integration of one or more XFL-based specifications in a closed-loop system simulating the global behavior of the fuzzy element(s) and their operational context. This simulation module offers an interface to specify:

- The variables that define the state of the closed-loop system, the simulation time and the end condition.
- The behavior of the closed-loop system under simulation, by means of the interaction of different elements, which can be XFL-based fuzzy specifications, any C-source module, or data stored in a file.
- The desired outputs from the simulation, as combinations of system variables.
- A set of parameters controlling the building and execution of the simulation process.

From these definitions, `xfsim` produces an executable file that can be initialized by the user and that runs under the control of the Xfuzzy environment. Once the simulation has finished, results are shown by starting different output viewers. Currently, two output formats are supported: a textual format that can be viewed by means of any text editor, and a gnuplot-compatible format.

4.2. Tuning XFL-based Systems

The module `xfbpa` tunes the parameters of an XFL-based fuzzy system by using supervised learning algorithms that employ the value of an error function defined for the system.

This learning module allows the user to fit the error function, giving more weight to those output variables considered as more significant in system deviation. Furthermore, it can apply a rule pruning algorithm, deleting those rules whose activation grade is below a (user configurable) threshold.

The user can select the algorithm to be applied among classical backpropagation, backpropagation with adaptive step size, Manhattan-rule based, R-prop [4], and quick-prop [5]. Calculation of the partial derivatives used by the learning algorithms are performed at runtime by the module.

Specific learning is also implemented since the user can select those variables, membership functions or parameters to be tuned. This way, different learning strategies can be employed, selecting the more adequate one for each particular problem.

The user interface permits to configure and execute the learning process, as well as to dynamically change its configuration and to survey its evolution. Monitoring facilities include a real time graphical representation of the system (mean and maximum) deviation.

5. Fuzzy System Synthesis

The Xfuzzy environment includes facilities for fuzzy system synthesis into software and hardware. The choice of the approach depends on both the complexity of the problem and its timing constraints. Software solutions offer great flexibility for defining the knowledge base, selecting fuzzy operators and adopting inference mechanisms. On the other hand, hardware solutions are required when problems demand high operational speed.

5.1. Software Synthesis

The obvious solution for software synthesis is the use of a standard and widely employed language like C. This allows software implementations produced by Xfuzzy to virtually run on all intended target architectures, from general purpose computers to microcontrollers.

The module `xfc` takes an XFL file and produces a C source file which implements the fuzzy inference engine defined in XFL as a C function. This file can be freely combined with any other modules to produce applications containing fuzzy logic capabilities. The interaction between the fuzzy inference engine and the calling module is performed through the parameters corresponding to the input/output variables of the fuzzy system.

5.2. Hardware Synthesis

Hardware realizations of fuzzy systems using the Xfuzzy environment can be accomplished following two strategies. In the first one, output values are precomputed off-line for all the possible input combinations and the inference process is carried out by a look-up table implemented as a combinational circuit [6]. Even though this approach allows a complete flexibility in the definition of the fuzzy system, its main drawback is the exponential growth of the required hardware resources when the number of inputs or the number of elements in the universe of discourse increase. To eliminate this handicap, the second approach included in Xfuzzy follows an on-line strategy where dedicated hardware is implemented to evaluate the inference process concurrently with the input changes [7].

Xfuzzy modules for hardware synthesis support two different implementation techniques. The use of FPGAs provides a fast prototyping capability. Besides, systems built with FPGAs exhibit intrinsic programmability, thus providing a simple mechanism to change or adjust their functionality. On the other hand, implementations as ASICs are more efficient in terms of silicon area and inference speed when the number of units to be fabricated is high.

The design flow for hardware implementation of fuzzy systems using Xfuzzy is illustrated in Fig 3. The starting
The point of the synthesis process is a behavioral description of the system using the specification language XFL. The verification process is carried out with the help of the simulation and learning tools provided by the environment. Once the system specification is validated, the designer can choose among three target implementations.

If a look-up table implementation is selected (right branch in Fig 3) the module xftl receives as input the XFL description and translates it to an input-output table with Berkeley’s PLA format. The PLA obtained is minimized and the boolean equations are extracted by using any compatible logic synthesis tool. These equations are then implemented by the logic blocks of a previously selected FPGA. In our case, the module xftl generates script files for the FPGA synthesis tools of Synopsys and Xilinx.

Two alternative implementations based on dedicated hardware can be accomplished by following the left branch in Fig 3. The module xfvhdl reads an XFL specification and generates a synthesizable VHDL description based on a specific architecture for fuzzy systems (see [8] for a detailed description of this architecture). Architectural options, related with the strategy used to store the knowledge base, and the number of bits of precision are defined by the user when xfvhdl is run. The module xfvhdl uses a cell library containing the parameterized VHDL description of the basic building blocks that make up the fuzzy system. The code used in this library is compatible with the restricted VHDL implementations of the Synopsys and Mentor Graphics tools. The module xfvhdl produces as output the following files describing the system:

- The system file, including the structural VHDL system description.
- Package files, containing the constants used in the VHDL description and the entities which define the system blocks.
- Knowledge base files, with information about antecedents, consequents, and rules.
- A testbench file, to ease the verification of the implemented system.

According to cost or timing constraints the user can choose to build the system as an ASIC or an FPGA. In the last case xfvhdl also generates script files to drive the Synopsys and Xilinx synthesis processes.

---

**Fig. 3: Design flow for hardware implementation in Xfuzzy.**
6. Xfuzzy Design Examples

In order to illustrate the versatility of the environment, and to analyse the application domains for the different hardware implementation techniques, we will address in this section the realization of fuzzy systems performing as function approximators.

The target function \( z=0.5(1+\sin(2\pi x)\cos(2\pi y)) \) has been studied as an example. We have explored the design space considering two implementation techniques (look-up table, and dedicated hardware using arithmetic MFC’s [9]) and two defuzzification methods (Fuzzy Mean and Weighted Fuzzy Mean [10]). A precision range from four to fourteen bits has been analysed in all the cases.

The rule base and parameters of the resulting fuzzy approximator (shown in Fig. 4) have been identified and adjusted by Xfuzzy. The system employs seven triangular membership functions to cover the input universes of discourse, and five different values \( (z_1, ..., z_5) \) for the consequents. Although consequents are represented in Fig 4a as bell-shaped functions, only one or two significant parameters are required for simplified defuzzification methods: in the Fuzzy Mean method (FM), the weight parameters are the bell function centers, while in the Weighted Fuzzy Mean (WFM), the second weight parameters correspond to bell function widths. The approximated surface is shown in Fig 4b.

The XFL descriptions were introduced as input files for the synthesis elements. All the approximators were designed using Xilinx XC4000 FPGA family. To evaluate the performance of the different hardware realizations, two characteristics have been considered: implementation cost in terms of the number of combinational logic blocks (CLBs), and approximation accuracy measured by the root mean square error (RMSE).

Fig 5a shows the cost evolution of the two implementation techniques as the number of bits increase. The look-up table approach provides the best results for low resolution implementations. Conversely, the dedicated hardware tech-
nique achieves better results as soon as resolution is more than 5 bits. Regarding the defuzzification methods for look-up table techniques, the cost of WFM is slightly higher than that of FM. However, dedicated hardware implementation of WFM requires one additional multiplier, thus increasing the cost compared to FM.

Fig 5b represents the RMSE as a function of the number of bits. The behavior is similar for the two implementation techniques. For low resolutions (up to 6 bits) the truncation error due to limited bus width is the dominant factor. When resolution is increased, the inherent fuzzy approximation error becomes the main error source. The results obtained let us analyse how the defuzzification method influences the RMSE. As shown in Fig 5b, WFM usually provides better results than FM. The difference between both defuzzification methods is more significant for the look-up table approach.

7. Conclusions

This paper presents the main aspects of Xfuzzy, a design environment which eases the description, verification and synthesis of fuzzy systems. Its more relevant features are the use of a simple but flexible specification language, XFL, and the inclusion of automatic synthesis tools that generate the hardware or software implementation of the system from a high-level behavioral description. The applicability of the different tools integrated into the environment has been analysed by exploring the design space for a function approximation example. Using this example some results have been obtained about hardware implementation techniques and simplified defuzzification methods.

References


