

Developer Toolkit for Embedded Fuzzy System Based on E-Fuzz

C. Chantrapornchai, K. Sripanomwan, O. Chaowalit, and J. Pipatpaisarn

Department of Computing, Faculty of Science,
Silpakorn University, Thailand 73000
ctana@su.ac.th

Abstract. In this work, we propose a development toolkit, called E-Fuzz-Wizard to help fuzzy system designers for designing embedded fuzzy systems. The toolkit composes of software and hardware that enables creating the rapid prototype. It contains the examples which use the hardware and code generated to produce a prototype. The software has a visual interface which allows the user to specify the requirement of fuzzy systems in terms of the fuzzy set characteristics, inference methods, rules and defuzzification method. It generates the code in C that is runnable in the chosen microcontroller platform. E-Fuzz Wizard also integrates unique features such as concurrent and real-time fuzzy system design as well as hardware mapping and customization. The generated code will facilitate the embedded fuzzy system development process. The toolkit is easy to use and facilitate the beginners to develop a fuzzy system.

Keywords: Embedded Systems, E-Fuzz, Fuzzy Design Tools.

1 Introduction

In Thailand, fuzzy system development is really required knowledge from experts in the field such as fuzzy controls. Also, to learn about the fuzzy systems, it is difficult for Thai students and teachers in high school to understand the use of fuzzy logic in every day life. According to the education policy by the government, it is urged to develop embedded system experts in the country. Many secondary schools and universities participate in the competitions related to embedded system fields in any platform. Fuzzy logic is a means to control many embedded equipments. It would be necessary to promote the fuzzy knowledge in such a field so that the domain experts will be expanded. It is found that even the teachers in high schools do not know fuzzy logic and its benefits. It is also difficult for them to understand in a short period. To help them understand better to gain more practical knowledge, it would be benefit if we have a laboratory fuzzy toolkit for the students to learn fuzzy logic development and its practices in the embedded world.

Fuzzy systems are now being used in many consumer electronic devices such as air conditions, washing machines, refrigerators etc. To implement a fuzzy system, one may choose to implement using many means such as general-purposed processors, fuzzy processors, programmable devices etc. In consumer electronics, the cost of the

device is important. Microcontrollers are the good choice since they are convenient to find and not expensive. However, the capability of the microcontrollers is very limited. Further programming on microcontrollers needs skills at the low level. Thus, together with complication in fuzzy systems itself, it is not very easy and convenient to create such systems on microcontrollers.

To develop a fuzzy control system, one needs to go through many steps to find the right parameters. These parameters are membership functions, rules, defuzzification methods, and inference methods. A lot of software simulations need to be done to select the proper parameters. Each set of parameters yield distinct system characteristics, e.g., static and dynamic memory usage, execution speed, and accuracy. After the parameters are selected, the fuzzy system is then implemented on the particular hardware[4].

The objective of this research is to develop a toolkit to help building the rapid prototype of simple embedded fuzzy applications. Also, the toolkit should facilitate the study of fuzzy system development in the laboratory which shows the practice of embedded fuzzy systems. The toolkit contains the hardware, software, and laboratory manual. The software is called E-Fuzz Wizard. It has a visual interface in a drag and drop manner. The user specifies the fuzzy system requirement, i.e., the membership function shape, the inference methods, rules, and defuzzification method. Then the code in C for the specified platform is generated. The tool also includes the simulator which enables the testing of the fuzzy requirements before real implementation. Besides, the tool supports the concurrent and real-time features which allow the more subsystems to be run and communicate concurrently. It allows the user to specify the mapping on the physical device to view the testing results when applying the hardware. The tool then generates C fuzzy application prototypes on microcontrollers as well as FPGA. The hardware sample and the laboratory manual guide the sample development of typical prototype fuzzy systems.

Several works have been done in developing fuzzy control hardware. Most fuzzy processors often have limitation such as 2 inputs and 1 output rules [7,9,22], the shape of the membership is triangular or trapezoid. Some work is based on analogue systems [10,15,17]. Some are specific to applications e.g. pattern recognition [20]. Some requires extra hardware supports and special instruction sets [19]. Many works are based on VLSI systems such as [16,21]. Ascia and Catania [3] presented a VLSI hardware for fuzzy processing. The hardware can handle 8 inputs, 4 outputs and 256 rules. Chen et.al. [5] proposed a high speed parameterized fuzzy processor. The processor considers parallel and pipeline processing. A general-purposed CPU is another choice which is flexible but may be too much powerful. Microcontrollers often provide a moderate solution since it is programmable and easy to create a prototype. However, many computations in fuzzy systems are expensive, such as the use of floating point, multiplication and division etc. They are not suitable to the 8-bit microcontroller unless certain optimization is done.

Many previous works have mentioned about the fuzzy development software and tools. Nishidai and Hajimi [12] present a tool which consists of hardware for fuzzy rule reasoning. Nishiuchi and Masamitsu [3] presented an approach to generate code for a fuzzy control program. Ahmed et.al. [1] presented an adaptive fuzzy software management tool. The tool can cope with fuzziness in the software development

process. Iqbal et.al. [2] proposed a fuzzy expert system for a manufacturing process, particularly in a machining process. The expert system can adapt and learn automatically. Zhang and Kandel [23] proposed a CPU scheduling method using fuzzy logic under a multiple criteria. Mateou and Andreou [11] presents a framework to develop decision support systems which uses fuzzy cognitive maps and genetic algorithms. Rasmussen and Yager [14] developed a fuzzy query language called SummarySQL which is able to perform a smart query and search for data mining. One of the tools that are close to ours is fuzzyTECH. It is the commercial work of fuzzyTECH which provides fuzzy libraries. Their target platform is based on MCS51 and MCS96[25]. Xfuzzy is another well-known software tools which enable the developments of fuzzy[29]. It provides the multiple fuzzy system definitions and generates the code in C/VHDL. Though both works are applicable, it does not have an explicit support for the concurrent features.

Using our tool, it is convenient to understand the fuzzy system development. The visual software aids the system design. The users only put the whole system architecture and define each system's characteristics. Then the simulation helps view the behavior of the systems. Once the design is settled, the code implementation is generated. It can be programmed in the given hardware. Once hardware is programmed, the system is ready to use.

Our tool has special characteristics:

In the software, we overcome the limitation of existing ones such as the fixed number of inputs and the limited shape form of the membership function. It provides a flexible interface which allows designers to create many system inputs, the shapes can be varied, and many of multiple-input rules may be used. Using the tool, developers can learn to also create a more complex system by adding the concurrent real-time features. Such features are unique and not commonly found in any fuzzy expert system design tool. The code generated is in C on PIC microcontrollers with the good estimation of memory usages and timing characteristics.

The hardware included are PIC microcontroller board whose CPU can be replaced. The designer can use their own PIC CPU or the given one in the set.

The code generated is programmed by any existing IDE such as MPLAB or NIOS II. Thus, the hardware platform can be flexible. The given hardware in the set is for convenience in developing systems the student laboratory purpose.

The paper is organized as following: next section presents fundamental fuzzy system components and the inference process and related computation. Section 3 presents the structure of E-Fuzz Wizard and its capabilities. Section 4 presents an example of UI interface and development methods using the tool. Section 5 shows the application examples. Section 6 concludes the paper and discusses the future work.

2 Backgrounds

Typical fuzzy systems have the characteristics as shown in Figure 1.

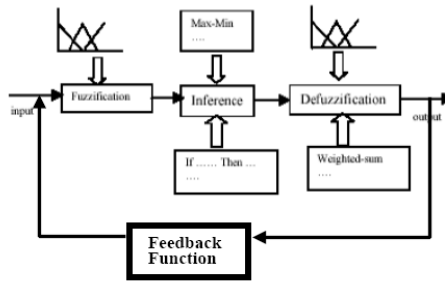


Fig. 1. Fuzzy System Components

In Figure 1, a given input is read and then fuzzified to be a fuzzy set. This is called the fuzzification process. After that, the value is given to the inference engine to find out which rules are fired. For each rule that is fired, the corresponding output linguistic variable is marked. This step is called a fuzzy inference process. Then, all the fired output linguistic values are concluded to be a crisp value which is the actual output. This is called the defuzzification process. The output is given to the feedback function which is a computation of some linear/nonlinear function and it becomes the input again.

Based on these steps, necessary parameters that one needs to come up with when the system is designed are the input and output variables, linguistic variables for each input and output and their membership functions, fuzzy rules, inference method and defuzzification method.

In the following, we briefly explain steps to establish a fuzzy system according to the above components [18].

1. Inputs and outputs: First designers need to define the number of inputs and output. The universes of each input and output are defined. This defines domain for each fuzzy set.

2. Linguistic variables and membership functions: For each input and output, one needs to define the set of linguistic variables. Each linguistic variable corresponds to a membership function which specifies a mapping from an element to a degree of membership value ranged $[0,1]$. The membership function is typically defined by triangular shape, trapezoidal shape, bell shape, etc.

3. Rule set: From the given inputs and output and linguistic variables for each one, the set of rules are defined. Typically, the rules are defined from all possible combinations of input linguistic variables. After that, the rule optimization may be done to minimize the number of rules, the number of inputs for each rule, and to minimize the memory usage for the rule set. Most fuzzy systems require 2 inputs and one output. Many fuzzy hardware implements the 2-dimensional associative memory for the rule set.

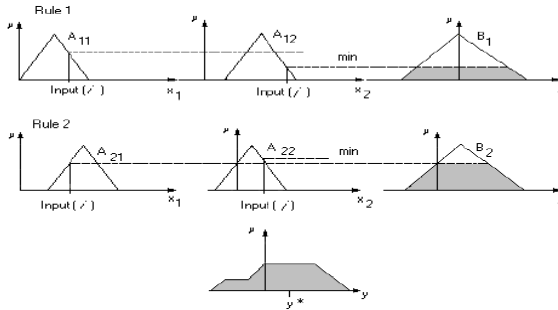


Fig. 2. Max-min inference

4. Inference method: Designers need to specify the fuzzy inference method used to compute the output membership degree. Many operators are defined in the literature [23]. The commonly used one is max-min or max-product. Figure 2 shows a graphical example of using max-min inference[18]. The minimum value between two membership values of the two inputs are used as the cut to the output linguistic variable for each rule. Then the fuzzy set for each output linguistic variable is unioned to become a final set and the final set is defuzzified. For max-product, rather using min operation, the product operation is used instead.

$$z^* = \frac{\int \mu_C(z) \cdot z dz}{\int \mu_C(z) dz} \tag{1}$$

5. Defuzzification method: From Figure 2, the final set is defuzzified to get a crisp output value y^* . Several methods can be used to defuzzify such as using the max value (means of max, smallest of max, largest of max), computing a centroid value, using the approaches such as weighted-average, center of sum. The commonly used one is centroid which is computed as the equation (1) and is depicted by Figure 3.

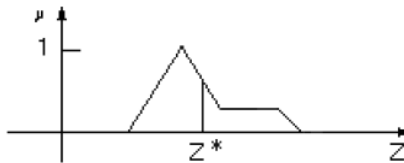


Fig. 3. Centroid defuzzification

Another example of the simpler defuzzification method is weighted-average. This is described by the equation (2) and Figure 4.

$$z^* = \frac{\sum \mu_C(\bar{z}) \cdot \bar{z}}{\sum \mu_C(\bar{z})} \tag{2}$$

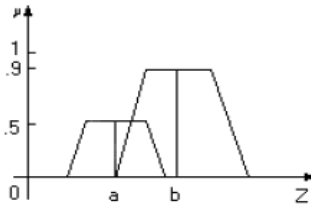


Fig. 4. Weighed-average defuzzification

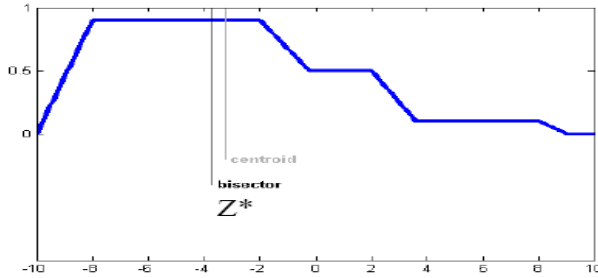


Fig. 5. Centroid VS Bisector

Another example for defuzzification method which has a good performance is bisector approach. The approach computes the vertical line z^* which divides into two equal regions. Figure 5 compare the points obtained by centroid and bisector [28].

3 E-Fuzz Software and Components

In E-Fuzz, the target architectures are FPGA and PIC microcontrollers. The current version of the software focuses on the microcontroller. Figure 6 shows the E-Fuzz menu for a user to develop a particular fuzzy system.

File	Edit	Hw Setting Mode	Fuzzy System Config.
New Project	Cut	<input type="checkbox"/> PIC Micro Controller	Select PIC no.
Open Project	Copy		Customized PIC
Save	Paste	<input type="checkbox"/> FPGA	
Save As	Delete		
Exit			
			Sub System
			New Sub System
			Set timing
			Set Input Linguistic Data
			New Input Linguistic Data
			Edit Input Linguistic Data
			Set Output Linguistic Data
			New Output Linguistic Data
			Edit Output Linguistic Data
			Set Rule
			Test
			Crisp Test
			Fuzzy Test
			Set Inference and Defuzzification method
			Map variable port HW
			Add virtual port
			Generate Code

Fig. 6. E-Fuzz Menu

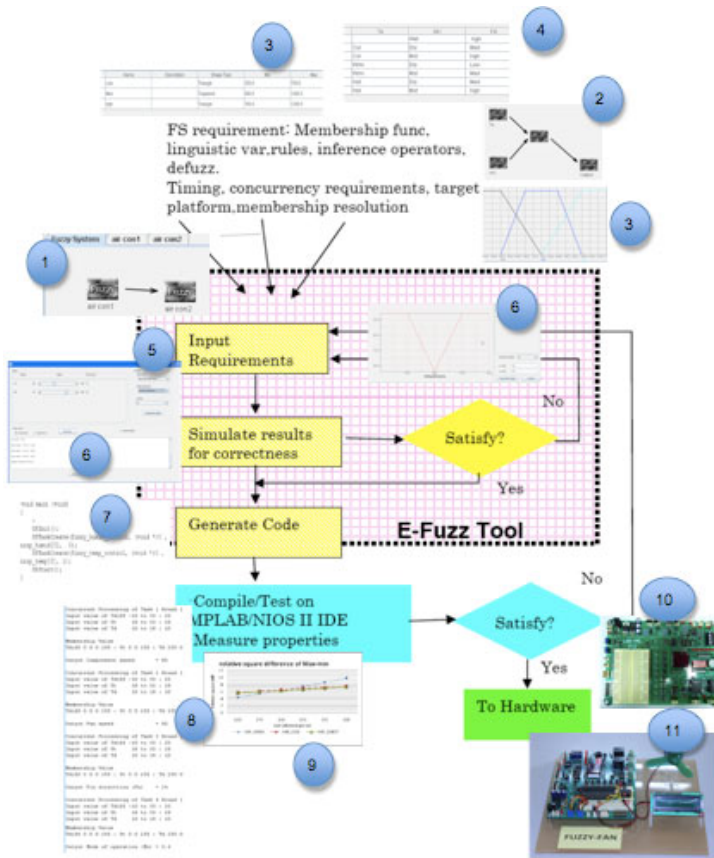


Fig. 7. Usage of E-Fuzz Tool

From E-Fuzz Components in Figure 6, a user needs to specify the overall architecture of the system first. He specifies the details of fuzzy systems and their interconnections. Each fuzzy system is defined based on the given parameters such as input/output domain, linguistic variables, inference method, defuzzification method, etc. Each system can be simulated by its own using either crisp or fuzzy input test against the given feedback function. The code for each fuzzy system can then be generated in the specified platform after the user satisfies with the parameter setting. For the overall system, the communications between fuzzy systems can be given as output-input relation. As the hardware is specified, the port mapping of input/output of fuzzy systems to the real device port or virtual port can be done. Virtual ports are defined by the operating system level such as mailbox and queue in Micro C/OS II. Each fuzzy system may be run periodically. Inputs may also be read periodically. Once these timing properties are given and the period is specified, the main code containing periodic fuzzy systems and function codes are generated.

Figure 7 shows the framework of using the software. The developer defines the system architecture. (1) displays the whole system. (2) displays each system architecture.

Then each system parameter are defined: (3) inputs, output, linguistic variables, (4) rules, (5) inference method and defuzzification. After that, he verifies the behavior of the system in the tool (6). Once the behavior is satisfied, the code implementation can be generated automatically (7). Then the code is programmed using the existing tools (8). The code properties may be verified again in these tools (9). Since C code is in a high and each IDE has a specific compiler, the code low-level property depends on it. The developer may need to tune the code to adjust in details. Then the code is programmed into the board in the toolkit (10) and accessories are connected for practice (11). In the testing, we use MPLAB (www.microchip.com) or NIOS II (www.altera.com) since we are interested in developing the prototype in PIC and FPGA. Each of these components phase will be described in the next sections.

4 Case Study Dialogs

Figure 8 shows the case where we design a system containing two connected subsystem. On the edge, we can specify a way to communicate the output as an input on another system.



Fig. 8. Main Diagram

Then the user specifies each fuzzy system component’s properties. This is as shown in Figure 9. In Figure 9, we develop a fuzzy temperature control. The system needs two inputs which are humidity and temperature values and gives one output which is a fan speed setting.

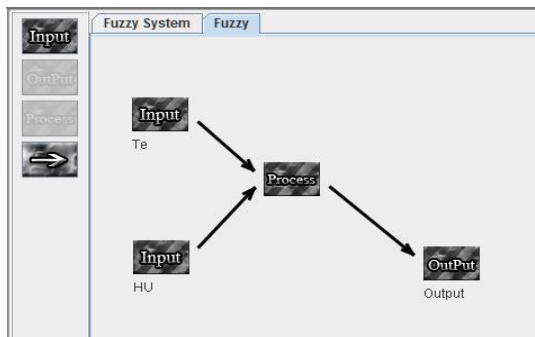


Fig. 9. Specification of fuzzy components


```

void main (void)
{
    :
    OSInit();
    OSTaskCreate(fuzzy_humid_control, (void *)0 ,
&inp_humid[0], 0);
    OSTaskCreate(fuzzy_temp_control, (void *)0 ,
&inp_temp[0], 1);
    OSStart();
}

```

Fig. 10. Main code for concurrent fuzzy systems

After all the specification is done, the code is generated. Figure 10 shows the pseudo code of the generated code for concurrent systems. The code to generate is pretty much straightforward.

```

void fuzzy_humid_control (void *pdata) {
    while (1) {
        Fuzzy_humid();
        OSTimeDlyHMSM(0,0,1,0); // delay 1 second
    }
}

```

Fig. 11. Fuzzy Task with delay

For this case we develop two fuzzy systems. We use Micro C/OS II to support the concurrency fuzzy systems. Each system is defined as a task in Micro C/OS II. Each fuzzy task is implemented in a loop with some delay as a period in Figure 11. It is shown that each fuzzy system in the diagram is mapped to a function corresponding to a task. The communication between fuzzy tasks is done by mail box or queue in Micro C/OS II. Using the library, the code functions necessary for fuzzy computation are reused.

5 Hardware Kit and Examples

In the toolkit, we also provide a set of hardware and its peripherals to help set up the fuzzy logic laboratory. The hardware board is based on PIC CPU. The board has the palette (420 holes) to stick the PIC processor on. It contains dip switches, 2 DC inputs which are suitable for connecting sensors, relay circuit, LED, serial port, reset circuit, RTC, 7-segment display, buzzer, D/A for testing typical examples in fuzzy controls. The LAB-PIC board is shown in Figure 12. The example of PIC that comes with the kit is shown in Figure 13. This one is PIC18F6627.

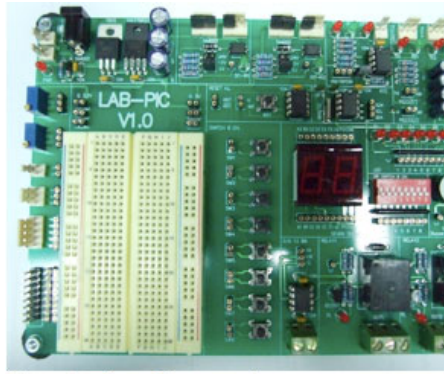


Fig. 12. LAB-PIC Board



Fig. 13. PIC chip example that is in the kit (PIC18F6627)

We also include the manuals which present easy steps for fuzzy design examples using the software to help learning fuzzy system developments. Based on the fuzzy fan example shown in the previous section, we simplify it and take the generated code and program into the LAB-PIC. We connect the input to the resistors assuming they are the sensor reading of both temperature and humidity (Figure 14). We also connect the output control to control the speed of the fan. The output is also displayed on the LCD shown in the figure. Figure 15 shows the LAB-PIC and hardware connection for testing.

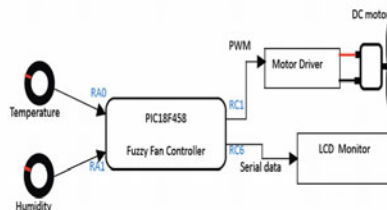


Fig. 14. Fuzzy fan structure

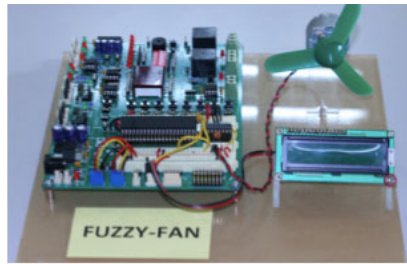


Fig. 15. Fuzzy fan example on LAB-PIC



Fig. 16. Fuzzy Inverted Pendulum example on LAB-PIC

We also have another example on an inverted pendulum [18]. In this example, we are to control the aluminum ruler. The input of the system is the angle read by the sensor (assuming it is the resistant value). The output of the system is the PWM value to control the direction to move the motor and to control the speed of the motor. The motor controls the standing of the aluminum ruler. The setup kit for this laboratory is shown in Figure 16.

Table 1. Comparison between E-Fuzz and others

Aspects	XFuzzy	FuzzTech5.7	E-Fuzz
Fuzzification-Membership function specification and Membership function code	-Specify graph parameters. Software draws the graphs. Support both curve and linear functions. -Generate as a function computing the value	-Drag graph for both curve, linear functions. N/A *	-Drag graph for both curve, linear functions. -Store points in array. Generate a function to map a value.

Table 1. (continued)

Rules Specification and Code	-Input as table, matrix, rule formats -Generate a code function with inference operator	-Input as rule, matrix formats N/A*	-Input as a rule format -Generate as code together with inference operators.
Defuzz.	CoG,MoM,LoM,FoM, Takagi, Weighted fuzzy mean, Gamma,Max	CoM,MoM	CoG,MOM,Bisect, LoM,Weight-average
Multiple systems/Concurrency	Yes Code generated individually for each step of fuzzy process as each function per system.	Yes N/A*	Yes Code generated can be run as tasks under Micro OS/II Tasks can be communicated using virtual using Mailbox/Queue or physical port of MCU.
Timing	No	No	-there is a timing associated with input readings and each system.
Language	C,C++,Java,VHDL	C .,Java,C++,VB,Matlab	C
Target platform	Not Specified	MCU,PLC	PIC, FPGA
Simulation/Debugging	Yes	Yes	Yes
Training support	Yes	Yes	No
Commercial	No	Yes	No

6 Conclusion

In this work, we present the toolkit for developing fuzzy systems. The kit contains 1) hardware 2) visual tool called E-Fuzz Wizard. 3) Sample laboratories. The hardware is designed containing standard interface and I/O used by simple control applications. The software provides the integrated facility to build the concurrent real-time and embedded fuzzy systems. It provides ways to specify each fuzzy system parameter visually. Real-time properties of the fuzzy systems can be specified. The software includes the simulation of various parameter setting of the fuzzy systems. Once the user satisfies with the parameter selection, the code for the target platform is generated. The current version targets at platforms : microcontroller PIC and FPGA. The generated code is written in C for microcontrollers with embedded real-time OS (micro C/OS II). The code is programmed to the hardware using existing programming IDE tools. The examples show a real practice of the designed fuzzy systems. All together, the kit gives an easy way to build the prototype fuzzy system for the beginners: from parameter selection, behavior tuning, code generation, until hardware mapping.

Acknowledgement

This work is supported in part by NECTEC, NSTDA, Thailand grant number NT-B-22-ES-17-51-01 and Faculty of Science Funding, Silpakorn University, Thailand.

References

- [1] Ahmed, M.A., Saliu, M.O., AlGhamdi, J.: Adaptive Fuzzy Logic-Based Framework For Software Development Effort Prediction. *Information and Software Technology* 47, 31–48 (2005)
- [2] Iqbal, A., Khan, I., Dar, N.U., He, N.: A Self-Developing Fuzzy Expert System, Designed for Optimization of Machining Process. In: *Proceedings of the World Congress on Engineering*, vol. III (2008)
- [3] Ascia, G., Catania, V.: An Efficient Hardware Architecture to Support Complex Fuzzy Reasoning. *International Journal on Artificial Intelligence Tools* 5(1-2), 41–60 (1996)
- [4] Chantrapornchai, C.: Rapid prototyping Methodology and Environment for Fuzzy Applications. In: *Optimization Techniques 1973*. LNCS, vol. 4, pp. 940–949. Springer, Heidelberg (2003)
- [5] Chen, B.T., Chen, Y.S., Hsu, W.H.: Performance evaluation of a parameterized fuzzy processor (FPF). *Fuzzy sets and systems* 81(3), 293–309 (1996)
- [6] Frías-Martínez, E.: Design of a Lukasiewicz rule-driven fuzzy processor. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 7(1), 65–71 (2002)
- [7] Gabrielli, E.G., Masetti, M.: Design of a family of VLSI high speed fuzzy processors. In: *IEEE Fuzz 1996* (1996)
- [8] Ghaus, C.: Fuzzy model and control of a fan-coil. *Energy and Buildings Journal* 33, 545–551 (2001)
- [9] Falchieri, D., Gabrielli, A., Gandolfi, E.: Very fast rate 2-input fuzzy processor for high energy physics. *Fuzzy Sets and Systems* 132, 261–272 (2002)
- [10] Li, J.H., Lim, M.H., Cao, Q.: Evolvable Fuzzy Hardware for Real-time Embedded Control in Packet Switching. *Evolvable Machines* 161, 205–227 (2005)
- [11] Mateou, N.H., Andreou, A.S.: A framework for developing intelligent decision support systems using evolutionary fuzzy cognitive maps. *Journal of Intelligent and Fuzzy Systems* 19(27), 151–170 (2008)
- [12] Nishidai, Hajimi: Fuzzy reasoning and methods, rule setting apparatus and methods. *Eurpoean Patent Classification* (1997): G06F 9/44. Publication number: EP0513829, <http://www.freepatentsonline.com/EP0513829.html>
- [13] Fumitaka, N., Masamitsu, I.: Method for generating fuzzy control program. *Japanese Patent no. JP7160306.3* (1995), <http://www.sumobrain.com/patents/jp/Method-generating-fuzzy-control-program/JP07160306.html>
- [14] Rasmussen, D., Yager, R.R.: SummarySQL - A Fuzzy Tool For Data Mining. *Intelligent Data Analysis* (1997)
- [15] Song, C.T.P., Quigley, S.F., Pammu, S.: Novel analogue fuzzy inference processor. In: *Proceedings of ISCAS*, vol. 3, pp. 247–250 (1998)
- [16] Pagni, A., et al.: Automatic Synthesis Analysis Implementation of a Fuzzy Controller. In: *IEEE Int'l Conf. Fuzzy Systems*, pp. 105–110. IEEE Press, Piscataway (1993)
- [17] Pammu, S.: Novel Analogue Fuzzy Inference Processor. In: *Proceedings of ISCAS*, vol. 3, pp. 247–250 (1998)

- [18] Ross, T.J.: Fuzzy Sets. In: Fuzzy Logic and Fuzzy Systems: Theory and Applications, McGraw Hill, New York (1995)
- [19] Salpura, V., Gschwind, M.: Hardware/Software Co-Design of a Fuzzy RISC Processor. Proceedings of the IEEE 83, 422–434 (1995)
- [20] Shi, B., Lin, G.: Programmable and expandable fuzzy processor for pattern recognition. United States Patent 6272476 (2001),
http://d.wanfangdata.com.cn/Periodical_dianzixb200002008.aspx
- [21] Masaki, T., Hiroyuki, W.: A VLSI implementation of a fuzzy inference engine: toward an expert system on a chip. International Journal on Information Sciences 38, 147–163 (1986)
- [22] Tsutomu, M.: Fuzzy processor, European Patent EP0392494 (1990)
- [23] Viot Greg, J., Sibigtrogth James, M., Broseghinl James, L.: A Method for performing a fuzzy logic operation in data processor. European Patent: EP0574714 (2000)
- [24] Zhang, Y.-Q., Kandel, A.: Fuzzy CPU Scheduling. International Journal on Artificial Intelligence Tools 6(2), 211–225 (1997)
- [25] <http://www.fuzzytech.com>
- [26] <http://www.cs.cmu.edu/afs/cs/project/ai-respository/ai/areas/fuzzy/systems/fuzzyfan>
- [27] <http://www.mathworks.de/products/demos/shipping/fuzzy/defuzzdm.html#3>
- [28] <http://www.micrium.com>
- [29] <http://www.imse.cnm.es/Xfuzzy>
- [30] <http://www.programmersheaven.com/download/1244/download.aspx>