E-Wizard Toolkit: A Prototype Tool for Concurrent and Embedded System Design

C. Chantrapornchai, K. Sripanomwan, O. Chaowalit
Department of Computing, Faculty of Science
Silpakorn University, Thailand
ctana@su.ac.th

Abstract

We propose a development tool, called E-Fuzz-Wizard to help design concurrent embedded fuzzy systems. It composes of three portions: software that enables the rapid prototype of fuzzy systems, the hardware prototype board, and the example kit . 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. The hardware is based on a PIC development kit for programming a prototype example. The example kit contains example code and demonstration hardware for learning fuzzy system development for embedded systems.

Keywords: *Fuzzy Design Tool, Visual Tool, Microcontroller, Rapid prototype, Embedded System.*

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.

3. Fuzzy Systems

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

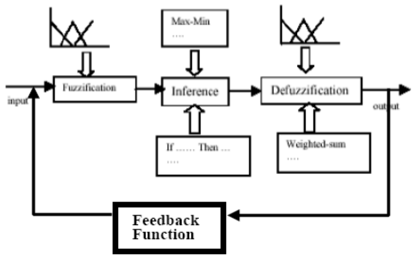


Figure 1.Fuzzy System Components.

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.

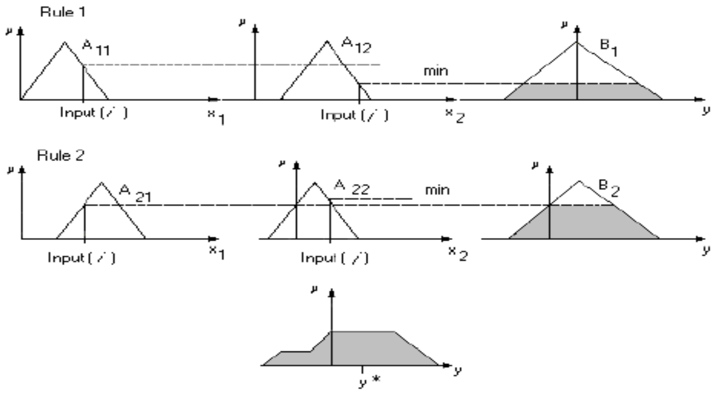


Figure 2. max-min Inference.

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.

$$z^* = \frac{\int \mu_{\underset{\sim}{C}}(z) \cdot z dz}{\int \mu_{\underset{\sim}{C}}(z) dz} \quad (1)$$

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 centriod 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.

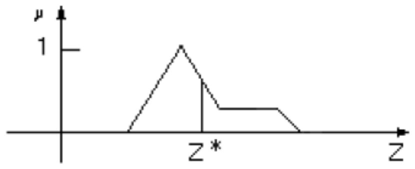


Figure 3. Centriod Defuzzification.

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

$$z^* = \frac{\sum \mu_{\underset{\sim}{C}}(\bar{z}) \cdot \bar{z}}{\sum \mu_{\underset{\sim}{C}}(\bar{z})} \quad (2)$$

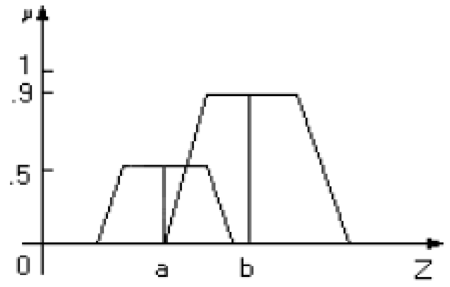


Figure 4. Weighed-Average Defuzzification

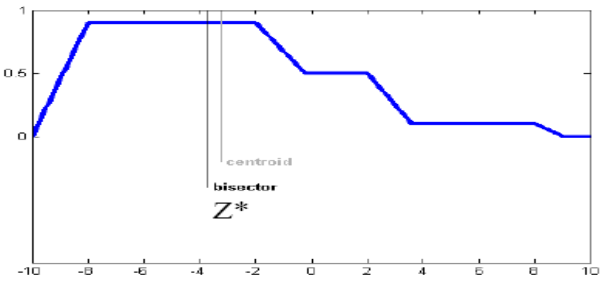


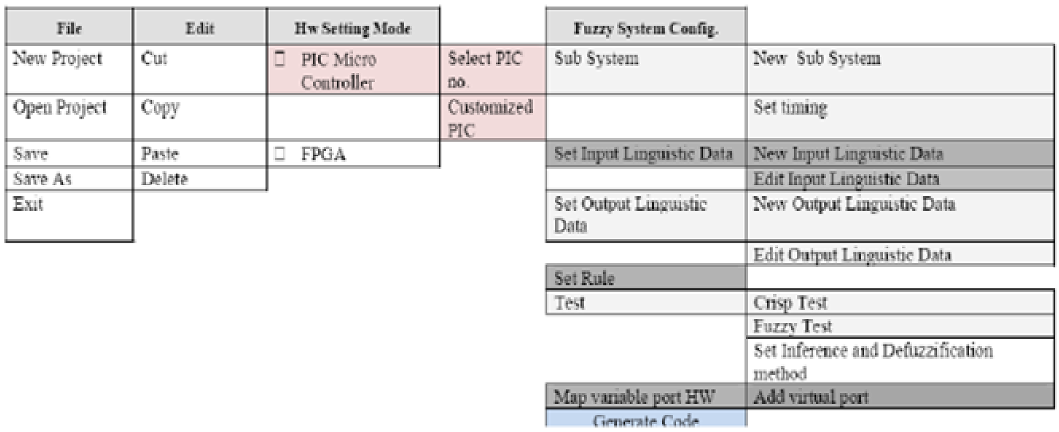
Figure 5. Centroid VS Bisector

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 centriod and bisector [28].

3. E-Fuzz Software 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.

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.



File	Edit	Hw Setting Mode		Fuzzy System Config.	
New Project	Cut	☐ PIC Micro Controller	Select PIC no.	Sub System	New Sub System
Open Project	Copy		Customized PIC		Set timing
Save	Paste	☐ FPGA		Set Input Linguistic Data	New Input Linguistic Data
Save As	Delete				Edit Input Linguistic Data
Exit				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	

Figure 6. E-fuzz Menu.

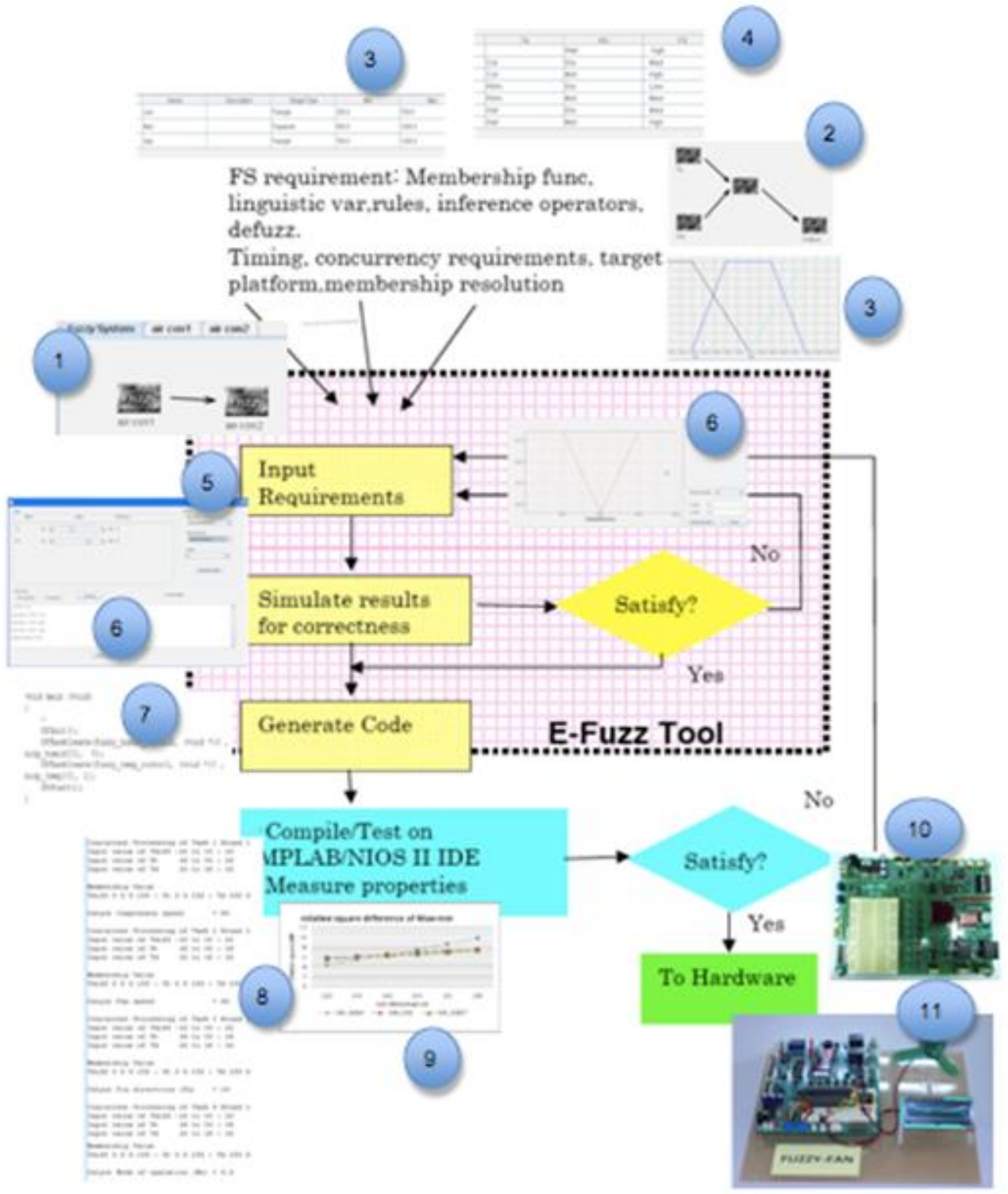

FS requirement: Membership func, linguistic var,rules, inference operators, defuzz.
Timing, concurrency requirements, target platform,membership resolution
Input Requirements
Simulate results for correctness
Satisfy?
No
Yes
Generate Code
E-Fuzz Tool
Compile/Test on MPLAB/NIOS II IDE Measure properties
Satisfy?
No
Yes
To Hardware
FUZZY-FAN
1
2
3
4
5
6
7
8
9
10
11

Figure 7. Usage of E-Fuzz Tool.

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 complier, 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.

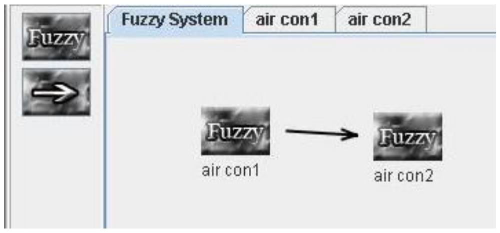


Figure 8. Main Diagram.

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.

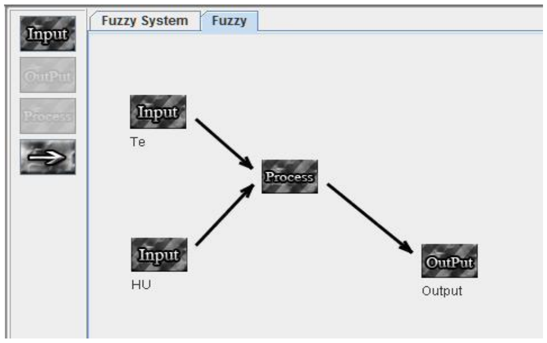
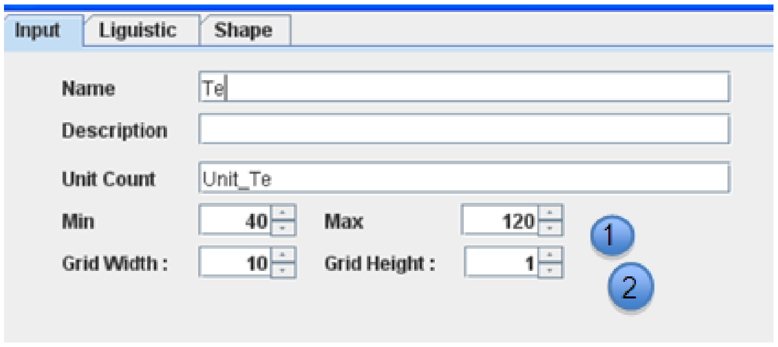


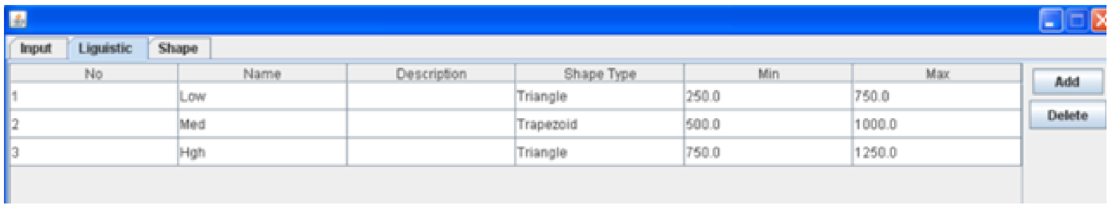
Figure 9. Specification of Fuzzy Components.

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. Figure 10 shows the input universe specification dialog of the temperature input. We specify the minimum and maximum values of the domain (1). We also specify the granularity of the domain by grid width and grid height (2). Figure 11 shows the linguistic variable specification of the temperature input. We define three linguistic variables whose shapes are defined on the column "shape type". Then Figure 12 shows the graph of each linguistic variable and the user can use the mouse to adjust the function shape. We repeat using these dialogs from Figure 10-Figure 12 for the second input and output variables. Then Figure 13 shows the rule base definition. We define based on the given linguistic variables for each input. After all, we perform the testing for the whole system. Figure 14 shows the testing dialog. The user selects the inference (1) and defuzzification (2) methods. Then the user adjusts each input value using the scroll bar (3). This is to perform the crisp input testing. Then by clicking the process button, the output values are shown in the text box region below (4). Figure 15 presents another option with fuzzy input testing. The user defines input fuzzy graph on both domains. The output value is shown in the textbox next to it. We also view the output result showing the relationship between input and output in Figure 16. Here we view the relationship of all temperature values and all output values. The temperature inputs are assumed for all values in the domain to generate the graph while the humidity value is fixed.



Name	Te
Description	
Unit Count	Unit_Te
Min	40
Max	120
Grid Width :	10
Grid Height :	1

Figure 10. Domain Inputs.



No	Name	Description	Shape Type	Min	Max
1	Low		Triangle	250.0	750.0
2	Med		Trapezoid	500.0	1000.0
3	Hgh		Triangle	750.0	1250.0

Figure 11. Linguistic Variable Definition.

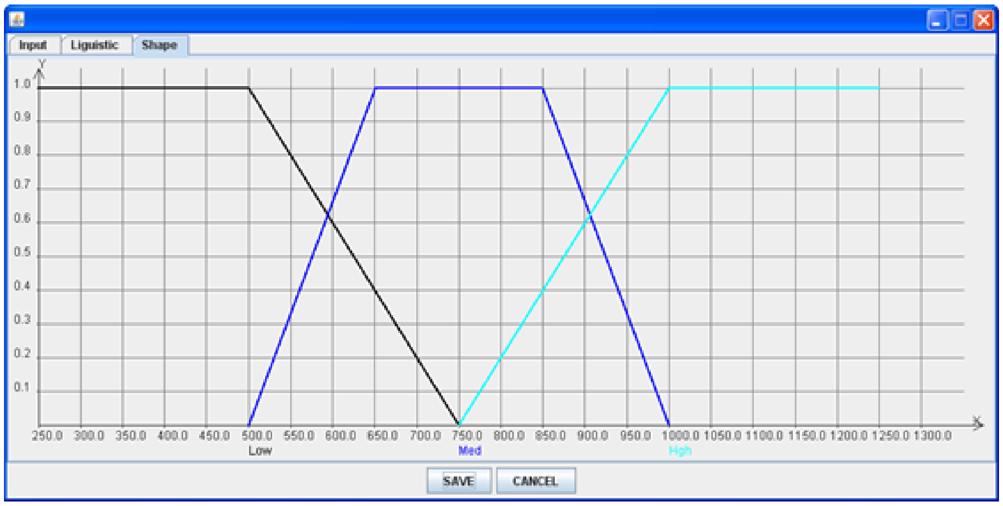
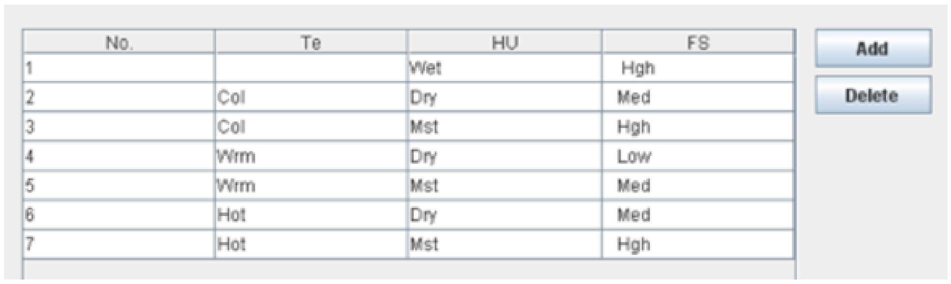
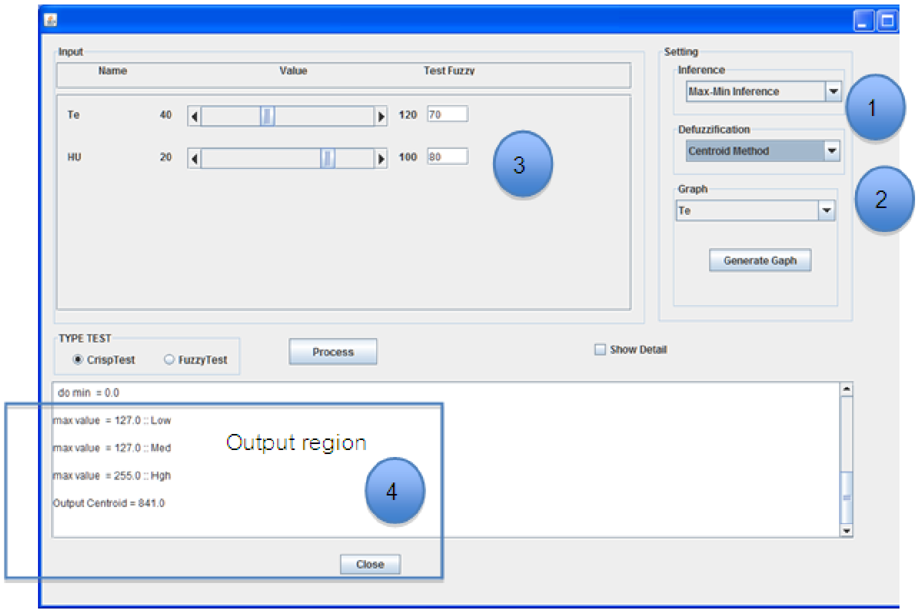


Figure 12. Membership Function for Each Linguistic Variable.



No.	Te	HU	FS
1		Wet	Hgh
2	Col	Dry	Med
3	Col	Mst	Hgh
4	Wrm	Dry	Low
5	Wrm	Mst	Med
6	Hot	Dry	Med
7	Hot	Mst	Hgh

Figure 13. Rule Definitions.



Input

Name	Value			Test Fuzzy
Te	40		120	70
HU	20		100	80

Setting
Inference: Max-Min Inference
Defuzzification: Centroid Method
Graph: Te

TYPE TEST: [✓ CrispTest] FuzzyTest

```
do min  = 0.0
max value  = 127.0 :: Low
max value  = 127.0 :: Med
max value  = 255.0 :: Hgh
Output Centroid = 841.0
```

Figure 14. Simulation Parameters and Output Dialog.

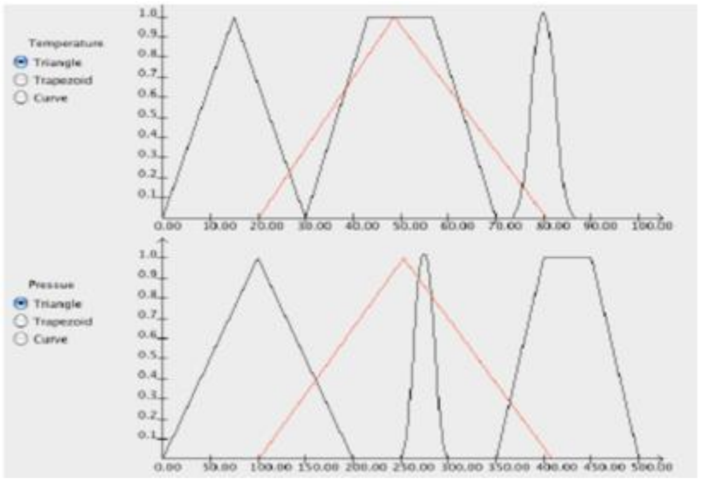
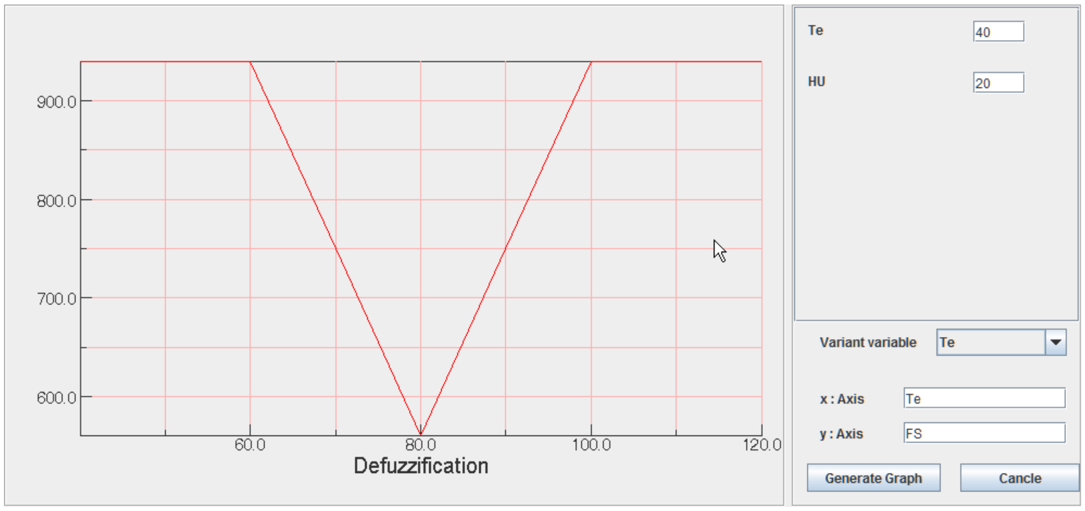


Figure 15. Fuzzy Input Testing.



Te 40
HU 20
Variant variable Te
x : Axis Te
y : Axis FS

Figure 16. Relationship of Input Temperature and Output Value.

```
main()
{
   code for reading initial inputs;
   while (1) {
    code for getting membership for each input
    code for inferences
    code for defuzzification
    code for computing next input using feedback function
   give some output to control something
              }
}
```

Figure 17. Pseudo Code for Main.

```
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();
}
```

Figure 18. Main Code for Concurrent Fuzzy Systems.

After all the specification is done, the code is generated. Figure 17 shows the pseudo code of the generated code for concurrent systems. The code to generate is pretty much straightforward and in Figure 18.

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 19. 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 mailbox or queue in Micro C/OS II. Using the library, the code functions necessary for fuzzy computation are reused.

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

Figure 19. Fuzzy Task with Delay.

5. Hardware Kit and Application 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 19. The example of PIC that comes with the kit is shown in Figure 21. This one is 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 22). 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 23 shows the LAB-PIC and hardware connection for testing.

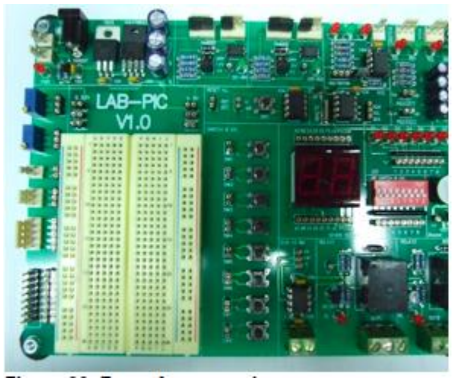


Figure 20. LAB PIC Board.

Figure 21. PIC Chip Example that is in the Kit (PIC18F6627).

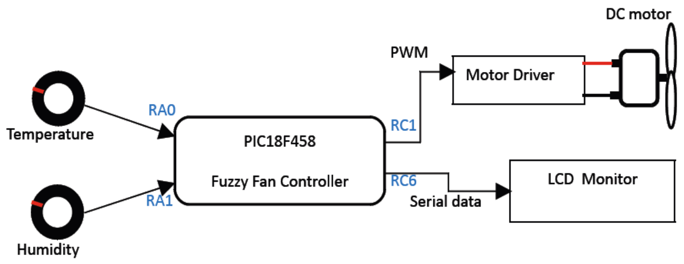


Figure22. Fuzzy Fan Structure.

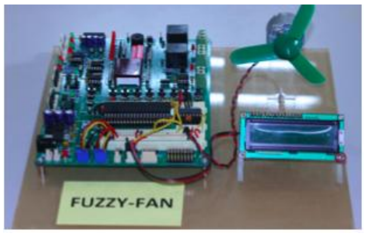


Figure 23. Fuzzy Fan Example on LAB-PIC.

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 24.

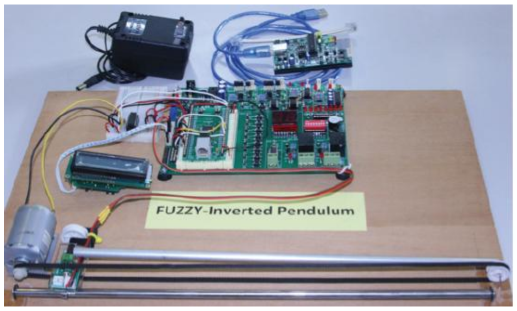


Figure 24. Fuzzy Inverted Pendulum Example on LAB-PIC.

These examples show how the kit is used to build the prototype in the following ways: For the hardware, 1) it has the palette to hold the PIC chip which can be replaced by any similar one. 2) It has standard input readings which may be connected to sensors which are commonly used for control applications. 3) It has standard outputs to view debugged results on LCD, LED and to control motor driving. For the software, 1) it has an easy to use user interface to build a fuzzy system. 2) It can handle various parameters in fuzzy system design. 3) It has a support to build concurrent fuzzy systems which are done by Micro C/OS II. This facilitate the learning of concurrent systems and fuzzy systems together. 4) the software contains the simulation which can simulate and view the fuzzy system results. 5) The code generated is editable and the programmable by existing IDE tools to the PIC/FPGA. We also test the concurrent system in DE2 board from Altera® FPGA. The emulation is done on Cyclone II as in Figure 25. The code for hardware display may be added to show the results.

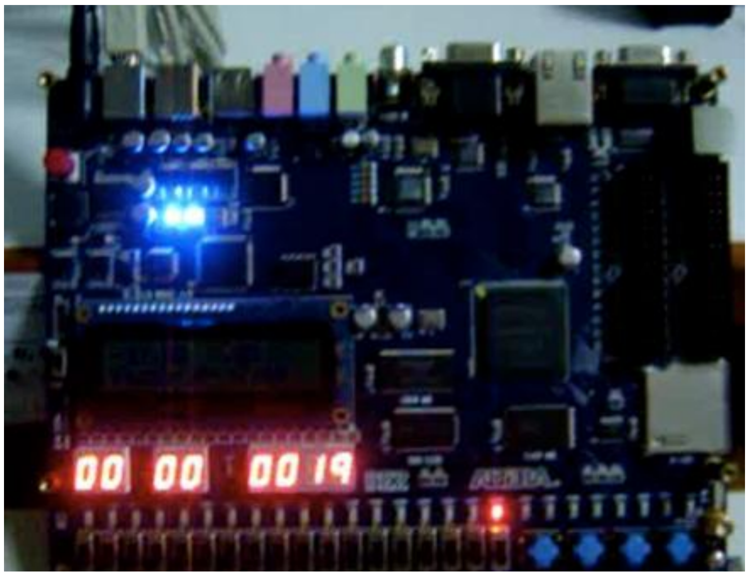


Figure 25. Running on DE2 Board from Altera ®.

We also test the code generated and measure the code properties for both timing and memory usages in previous work. The code generated is also tested against existing tool such as Xfuzzy, or standard C to check the correctness of the fuzzy system calculation. Table 1 summarizes the distinction in our tool and others. Note that our tool focuses on the code generation for PIC, concurrent supports by Micro C/OS II, and user specification by drag-drop approach.

Table 1. Comparison Between E-Fuzz and Others.

Aspects	XFuzzy	FuzzTyech5.7	FuzzGen1.6	E-Fuzz
Fuzzification -Membership function specification 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*	-Specify graph parameters, Software draws the graphs. Support linear function only. -Generate as a function computing the value	-Drag graph for both curve, linear functions. -Store points in array. Generate a function to map a value.
Rules – Specification – Code	- Input as table, matrix, rule formats -Generate a code function with inference operator	– Input as rule, matrix formats N/A*	-Input as rule formats.	-Input as a rule format -Generate as code together with inference operators.
Defuzzification	CoG,MoM,LoM,FoM, Takogi,Weighted fuzzy mean,Gamma,Max	CoM,MoM	No	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*	No	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	No	-there is a timing associated with input readings and each system.
Language	C,C++,Java,VHDL	C ,Java,C++,VB,Matlab	VB,Pascal,C,C++	C
Target platform	Not Specified	MCU,PLC	Not Specified	PIC, FPGA
Simulation/Debugging	Yes	Yes	No	Yes
Training support	Yes	Yes	No	No
Commercial	No	Yes	No	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, MA, Saliu, MO, and AlGhamdi, J, Adaptive Fuzzy Logic-Based Framework For Software Development Effort Prediction, *Information and Software Technology*, Vol.47 (Elsevier Science BV, 2005), pp. 31-48.

[2] Asif Iqbal, Iqbal Khan, Naeem Ullah Dar, and Ning He, A Self-Developing Fuzzy Expert System,Designed for Optimization of Machining Process, *Proceedings of the World Congress on Engineering* Vol. III (2008).

[3] Ascia G. and Catania V., An Efficient Hardware Architecture to Support Complex Fuzzy Reasoning, *International Journal on Artificial Intelligence Tool*s, Vol. 5(1-2) (1996), pp. 41-60.

[4] C. Chantrapornchai, Rapid prototyping Methodology and Environment for Fuzzy Applications. *Lecture Notes in Computer Science (ICCS 2003)*, Vol 4 (2003), pp. 940-949.

[5] Chen B. T, Chen Y. S, Hsu W. H., Performance evaluation of a parameterized fuzzy processor (PFP), *Fuzzy sets and systems*, Vol. 81(3) (1996), pp. 293-309.

[6] Frías-Martínez E., Design of a Lukasiewicz rule-driven fuzzy processor, *Soft Computing - A Fusion of Foundations, Methodologies and Applications*. Vol. 7 (1) (November, 2002), pp. 65-71.

[7] Gabrielli, E. Gandolfi and M. Masetti. Design of a family of VLSI high speed fuzzy processors. *IEEE Fuzz'96*

[8] Ghaus C. Fuzzy model and control of a fan-coil, *Energy and Buildings Journal* , Vol. 33 (6, 2001), pp. 545-

[9] Falchieri D., Gabrielli A., Gandolfi E. , Very fast rate 2-input fuzzy processor for high energy physics. Fuzzy Sets and Systems, Vol. 132 (2, 1 December 2002), pp. 261-272.

[10] Ju Hui Li, Meng Hiot Lim and Qi Cao, Evolvable Fuzzy Hardware for Real-time Embedded Control in Packet Switching , *Evolvable Machines,* Vol. 161(2005), pp.205-227.

[11] Mateou N.H and Andreou A.S. A framework for developing intelligent decision support systems using evolutionary fuzzy cognitive maps , *Journal of Intelligent and Fuzzy Systems*, Vol. 19 (27) (2008),

[12] Nishidai and 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] Nishiuchi Fumitaka and Ito Masamitsu. 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.and Yager R.R., SummarySQL - A Fuzzy Tool For Data Mining**,** *Intelligent Data Analysis*, (Elsevier Science, 1997).

[15] Song, C.T.P., Quigley, S.F., Pammu, S., Novel analogue fuzzy inference processor, *Proceedings of ISCAS*, Vol 3 (1998), pp.247-250.

[16] A. Pagni et. Al., Automatic Synthesis Analysis Implementation of a Fuzzy Controller, *IEEE Int'l Conf. Fuzzy Systems*, (1993) (IEEE Process, Piscataway, NJ), pp.105-110.

[17] Pammu, S. , Novel Analogue Fuzzy Inference Processor, *Proceedings of ISCAS*, Vol 3 (1998), pp. 247-250.

[18] Ross T.J*., Fuzzy Sets, Fuzzy Logic and Fuzzy Systems: Theory and Applications* (McGraw Hill,1995).

[19] Salpura V. and Gschwind M., Hardware/Software Co-Design of a Fuzzy RISC Processor. *Proceedings of the IEEE*, 83(3, March 1995) pp. 422-434.

[20] Bingxue Shi,Gu Lin. Programmable and expandable fuzzy processor for pattern recognition. United States Patent 6272476 (2001), http://d.wanfangdata.com.cn/Periodical_dianzixb200002008.aspx.

[21] Togai Masaki and Watanabe Hiroyuki, A VLSI implementation of a fuzzy inference engine: toward an expert system on a chip. *International Journal on Information Sciences*,Vol.38(2 April 1986.), pp. 147-163,

[22] Tsutomu, Miki, Fuzzy processor, European Patent EP0392494 (1990).

[23] Viot Greg J, Sibigtrogth James M., and Broseghinl James L, A Method for performing a fuzzy logic operation in data processor. European Patent: EP0574714 (2000)

[24] Yan-Qing Zhang, and Abraham Kandel, Fuzzy CPU Scheduling, *International Journal on Artificial Intelligence Tools*, Vol. 6 (2) (1997), pp. 211-225.

[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