Fifth IFSA World Congress (1993), 917 - 920

A General Approach to Encoding Heuristics on Programmable Logic Devices

J.Y. Leong, M.H. Lim and K.T. Lau

School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 2263
Tel: 065-799-5408 Fax: 065-791-2687
E-mail: emhlim@ntuvax.ntu.ac.sg

Abstract

Various forms of hardware alternatives exist for the implementation of fuzzy logic controllers. In this paper, we describe a systematic framework for realizing fuzzy heuristics on programmable-logic-devices. Our approach is suitable for the automated development of fuzzy logic controllers.

Keywords: Fuzzy logic, fuzzy controller, logic minimization and programmable logic devices.

1. Introduction

Various forms of hardware alternatives exist for the implementation of fuzzy logic controllers (FLC). Designers of fuzzy systems may opt to realize a FLC with decision making centred around a fuzzy inference processor [1]. Alternatively, one may consider using a general purpose microprocessor or microcontroller to execute the assembly codes of a FLC. Usually, the assembly codes are automatically generated from the fuzzy rules specified as high-level C programming codes. Each of the two approaches has its own merits and it is therefore not appropriate to assess qualitatively which is better. Generally speaking, the performance requirement (not necessarily the speed of inferencing) should be the major consideration in deciding the medium to realize a FLC.

Programmable-logic-devices (PLD) can be considered as another class of hardware alternative for realizing FLCs. We feel that the PLD approach offers the following unique advantages:

- For consumer products, PLD is an attractive alternative. Semi-custom programmable-logic-array (PLA) simplifies the electronics and the cost incurred is justifiable by the high production volume.
- PLD offers a sense of copyright protection. It is difficult to decode the fuzzy rules encoded into a PLD. Proprietary control strategy is therefore not so visible once programmed into a PLD.
- PLDs are capable of performing fuzzy inference at very high speed since every fuzzy logical inference is subjected only to access time delay.

In this paper, we outline a general approach for realizing FLCs on PLDs. We will first describe the overall approach from conceptualization of the fuzzy heuristics up to the point where the table for programming the FLC onto a PLD is generated. We illustrate our approach by means of the practical control example of a laundry machine by showing the steps involved before the final PLD table is achieved. The result of the actual control system based on the table generated will be described before concluding.

2. The Overall Approach

Figure 1 shows the stages of development from knowledge acquisition to implementation on silicon [3]. Initially, the control heuristics are captured by means of a fuzzy editor. The rules can be created using a normal text editor with the support of a fuzzy shell or it can be entered as a fuzzy associative mapping table. The functionality of the fuzzy rules and membership functions are verified using the fuzzy shell. Here, the process of tuning may occur until a set of well-tuned rules and membership functions are obtained.

The tuned knowledge is then converted to fuzzy relation matrices. The inferencing is performed by applying max-min composition operation to the fuzzified digital input generated by the event-sequencer and the matrices. Output corresponding to each input is converted to its digital representation by defuzzifying the fuzzy inference output. The resultant event look-up table basically consists of exhaustive digital input/output association.

The look-up table can be used for programming PLDs like ROM or EPROM and is not efficient for programming PLAs. For realization on PLAs, it is necessary that logic minimization is performed on the look-up table. The reduced form of the look-up table is more manageable for programming the AND/OR planes of PLAs. Typical reduction of 40% or more of the table is achievable after logic minimization.

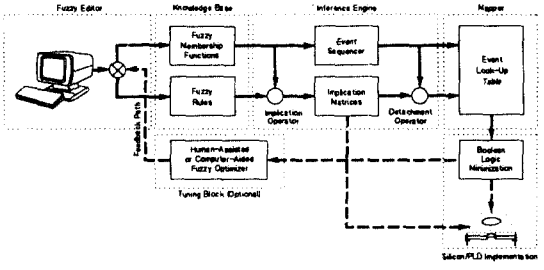


Figure 1: A framework for automated fuzzy systems implementation.

3. A Practical Example

In this section, we demonstrate the overall approach of implementating a FLC on PLD. The problem in consideration is a laundry process. The duration of a washing cycle will be determined based on two input variables. One variable is the degree of dirtiness in the fabrics and the other is the extent of greasiness. In actual washing machines, dirtiness and greasiness can be determined using optical sensors.

3.1 Knowledge Acquisition

A set of fuzzy linguistic descriptions or rules for describing the laundry process can be phrased as shown in Table 1. The corresponding membership functions of the fuzzy terms are shown graphically in Figure 2.

if	*<$greasiness is low_greasiness>*
and	*<$dirtiness is low_dirtiness>*
then	*<$time is very_short_time>*
else	
if	*<$greasiness is low_greasiness>*
and	*<$dirtiness is medium_dirtiness>*
then	*<$time is short_time>*
else	
if	*<$greasiness is low_greasiness>*
and	*<$dirtiness is high_dirtiness>*
then	*<$time is moderate_time>*
else	
if	*<$greasiness is medium_greasiness>*
and	*<$dirtiness is or (low_dirtiness, medium_dirtiness)>*
then	*<$time is moderate_time>*
else	
if	*<$greasiness is medium_greasiness>*
and	*<$dirtiness is high_dirtiness>*
then	*<$time is long_time>*
else	
if	*<$greasiness is high_greasiness>*
and	*<$dirtiness is or (medium_dirtiness, low_dirtiness)>*
then	*<$time is long_time>*
else	
if	*<$greasiness is high_greasiness>*
and	*<$dirtiness is high_dirtiness>*
then	*<$time is very_long_time>*

Table 1: Fuzzy linguistic rules for laundry machine.

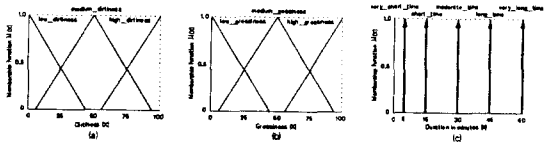


Figure 2: Membership functions for (a) $dirtiness, (b) $greasiness and (c) $time.

Each of the rule can be expressed as fuzzy relation. For example, the first rule can be represented by two fuzzy relations "low_greasiness → very_short_time" and "low_dirtiness → very_short_time". We will refer to the numerical representation of the fuzzy relation as implication matrix. Many methods of computing the implication matrix exist. The simplest and most commonly used is the cartesian product:

$$R_{A \to B} = A \times B \tag{1}$$

The membership values of the matrix can be computed as follows

$$\mu_R(u, v) = \min (\mu_A(u), \mu_B(v)) \tag{2}$$

where A and B are fuzzy subsets of the universe U and V respectively. For all the seven rules, 14 implication relations or matrices can be obtained. The implication matrices corresponding to the first rule are depicted in Table 2.

3.2 Composition Operation

This is illustrated by equation (3) where given an input A' and the implication matrix $A \to B$, we can compute the output B' by means of max-min composition.

$$\underset{A'}{[a_1 \ \dots \ a_m]} \circ \underset{A \to B}{\begin{bmatrix} b_{11} & \dots & b_{1n} \\ b_{21} & \dots & b_{2n} \\ & \vdots & \\ b_{m1} & \dots & b_{mn} \end{bmatrix}} \to \begin{bmatrix} c_{11} & \dots & c_{1n} \\ c_{21} & \dots & c_{2n} \\ & \vdots & \\ c_{m1} & \dots & c_{mn} \end{bmatrix} \to \underset{B'}{[d_1 \ \dots \ d_n]} \tag{3}$$

In equation (3), $c_{ij} = min\ (a_i, b_{ij})$ and $d_j = max\ (c_{1j}, c_{2j}, \dots, c_{mj})$ for $1 \le i \le m$ and $1 \le j \le n$. The operator "o" denotes fuzzy max-min composition.

0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

low_greasiness → very_short_time

0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

low_dirtiness → very_short_time

Table 2: Implication relations and matrices of a fuzzy rule.

3.3 Aggregation Operation

The inputs are matched against the *if*-part of every single rule. Each rule will fire accordingly, depending on the degree of matching between the inputs and the left-hand side of the rule. In addition, each rule, depending on how strongly it has fired, may affect the final conclusion to various extent. Therefore, a method of "consolidating" the conclusion deduced by each rule is required. This process is known as aggregation. In its simplest case, fuzzy aggregation can be achieved by a max operation. Table 3 outlines the fuzzy aggregated outputs for all input combinations. Each input nybble is associated with an element in the appropriate set of membership functions.

3.4 Defuzzification Process

It is common to use the centre of area to defuzzify a fuzzy conclusion. The centre of area can be computed as follows:

$$C.O.A. = \frac{\sum_{i=1}^{n} \mu_F(m_i) \cdot m_i}{\sum_{i=1}^{n} \mu_F(m_i)} \tag{4}$$

where n is the number of elements in the consequence universe which is characterized by the membership function μ_F and members m_i. A list

of defuzzified outputs can be obtained from the fuzzy aggregated outputs. For comparison, two different resolutions of defuzzified outputs are presented. The 4-bit defuzzified output represents the integer part of the centre of area calculated whereas the 8-bit counterpart reflects the integer part of the centre of area which is pre-multiplied by 10.

Referring to Table 4, it is not difficult to appreciate the benefits of having a less resoluted output. On the contrary, if crucial, the fractional part of the defuzzified output could be exploited to enhance specificity. To a greater extent, one might even try to increase the functional membership resolution for better accuracy. Nevertheless, it is up to the individual for the final implementation, bearing in mind that off-the-shelf PLA might not come in handy for odd bit size.

$greasiness	*$dirtiness*	*Fuzzy Aggregated $time*
0000	*0000*	*0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0*
0001	*0000*	*0.0 0.8 0.0 0.0 0.0 0.0 0.2 0.0 0.0 0.0*
0010	*0000*	*0.0 0.6 0.0 0.0 0.0 0.0 0.4 0.0 0.0 0.0*
0011	*0000*	*0.0 0.4 0.0 0.0 0.0 0.0 0.6 0.0 0.0 0.0*
0100	*0000*	*0.0 0.2 0.0 0.0 0.0 0.0 0.8 0.0 0.0 0.0*
0101	*0000*	*0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0*
⋮	⋮	⋮
0101	*1010*	*0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0*
0110	*1010*	*0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.8 0.0 0.2*
0111	*1010*	*0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.6 0.0 0.4*
1000	*1010*	*0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.0 0.6*
1001	*1010*	*0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.8*
1010	*1010*	*0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0*

Table 3: Aggregated outputs for all possible input combinations.

$greasiness	*$dirtiness*	*4-Bit Defuzzified $time*	*8-Bit Defuzzified $time*
0000	*0000*	*0010*	*00010100*
0001	*0000*	*0011*	*00011110*
0010	*0000*	*0100*	*00101000*
0011	*0000*	*0101*	*00110010*
0100	*0000*	*0110*	*00111100*
0101	*0000*	*0111*	*01000110*
⋮	⋮	⋮	⋮
0101	*1010*	*1010*	*01100100*
0110	*1010*	*1010*	*01100100*
0111	*1010*	*1011*	*01101110*
1000	*1010*	*1011*	*01101110*
1001	*1010*	*1100*	*01111000*
1010	*1010*	*1101*	*10000010*

Table 4: Defuzzified outputs for all possible input combinations.

3.5 Boolean Logic Minimization

The minimum PLD size required for programming Table 4 is at least *(4 + 4) × (11 × 11) × 4 bits* or *(4 + 4) × (11 × 11) × 8 bits* (*inputs × product terms × outputs*), depending on the precision of the defuzzified output selected. An ordinary ROM would more than satisfy the need if the speed of access is not critical. For PLA implementation, however, significant savings would be possible by adopting a less explicit output. PLAs are also preferred over ROMs or PALs because of the greater flexibility offered in configuring both the AND and OR planes.

However, the extent of logic minimization possible is case dependent and can varies between 40% to more than 70% of the original size. One contributing factor is the choice of an output resolution which can alter the final PLD size significantly. Table 5 shows the results of logic minimization done using the defuzzified outputs from Table 4. In comparison, Table 5 would dictate a PLA configuration of at least *8 × 75 × 4* or *8 × 75 × 8 bits* which is about *61.98%* of the original *8 × 121 × 4* or *8 × 121 × 8 bits*.

4. Application Results

A 3-D plot of the control surface which characterized the PLD FLC is given in Figure 3(a). The corresponding contour plot is also shown in Figure 3(b). The X-axis of the time-contour plot represents greasiness while the Y-axis denotes dirtiness. We can see from both plots that the solution space of the problem is well defined.

Qualitatively, no difference in performance would be observed between a conventional fuzzy controller and a fuzzy PLD controller of the same resolution, if given the assumption that both are able to provide the same inferencing throughput. However, this assumption could only be valid if a comparatively rapid fuzzy inference processing hardware strategy is available. In terms of speed, a rate of 50 MFLIPS (Fuzzy Logical Inferences Per Second) is achievable based on the Xilinx XC3000 FPGA implementation of the PLD [2]. A substantial speedup is possible when faster or superior versions of the Xilinx FPGA are used.

Product Term	*Input Variable $greasiness*	*$dirtiness*	*4-Bit Output Function*
1	*0000*	*000X*	*0010*
2	*0000*	*00X0*	*0010*
3	*0001*	*00XX*	*0011*
4	*000X*	*0011*	*0011*
⋮	⋮	⋮	⋮
72	*1000*	*1010*	*1011*
73	*1010*	*1001*	*1100*
74	*1001*	*1010*	*1100*
75	*1010*	*1010*	*1101*

Product Term	*Input Variable $greasiness*	*$dirtiness*	*8-Bit Output Function*
1	*0000*	*000X*	*00010100*
2	*0000*	*00X0*	*00010100*
3	*0001*	*00XX*	*00011110*
4	*000X*	*0011*	*00011110*
⋮	⋮	⋮	⋮
72	*1011*	*1011*	*01101110*
73	*1100*	*1011*	*01111000*
74	*1011*	*1100*	*01111000*
75	*1100*	*1100*	*10000010*

Table 5: Truth tables of defuzzified outputs after logic minimization.

5. Conclusions

Although an example of encoding fuzzy heuristics on PLDs was introduced, the possibilities remained for exploration are almost endless. Some opportunities include the implementation of custom fuzzy operators, selective integration of desired fuzzy modules on the PLDs as well as cascading of the PLDs to achieve parallel processing. Similar endeavours on the software environment is also highly pragmatic. In this paper, we demonstrate a systematic and versatile approach for implementing FLC using PLAs.

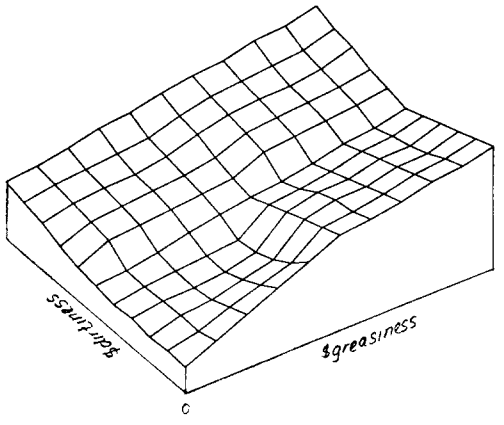
References

[1] M.H. Lim and Y. Takefuji, "Implementing Fuzzy Rule-Based Systems on Silicon Chips", IEEE Expert, February 1990, pp. 31-45.

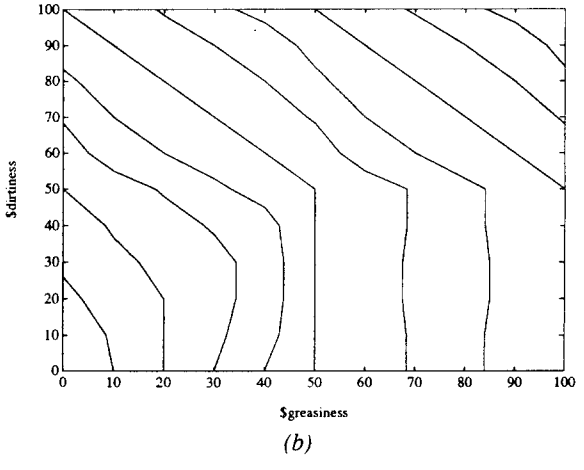
[2] M.A. Manzoul and D. Jayabharathi, "Fuzzy Controller on FPGA Chip", Proceedings of the First IEEE International

Conference on Fuzzy Systems, 1992, pp. 1309-1316.

[3] J.Y. Leong, M.H. Lim and K.T. Lau, "Development Strategy of Fuzzy Controller", Proceedings of the Second International Conference on Automation, Robotics and Computer Vision, Vol. 2, pp. CO.7.8.1-CO.7.8.4, 1992.



(a)



(b)

Figure 3: Control surface of the PLD FLC in (a) 3-D plot and (b) contour plot.