DESIGN AND APPLICATION OF DIGITAL FUZZY CONTROLLERS

S. Sánchez Solano, A. Barriga, C. J. Jiménez, J. L. Huertas

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

Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE’97),

© 1997 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author’s copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.
Design and Application of Digital Fuzzy Controllers

S. Sánchez-Solano, A. Barriga, C. J. Jiménez, J. L. Huertas
Instituto de Microelectrónica de Sevilla
Avda. Reina Mercedes, s/n Edif. CICA. E-41012 Sevilla, Spain
e-mail: santiago@cnm.us.es

Abstract

This paper focuses on hardware implementations of fuzzy inference systems which provide low silicon cost, high operational speed and adaptability to different application domains. The architecture and basic building blocks of two fuzzy logic controllers are described and their functionality is illustrated with experimental results showing the capability of these systems to be applied as function approximators.

1. Introduction

The application of fuzzy technologies to real-time control problems demands the development of new processing structures which allow efficient hardware implementations of fuzzy inference mechanisms. The solutions reported in the last years can be classified in two categories considering the implementation techniques that have been utilized. The use of analog techniques allows the realization of fuzzy chips with a good “silicon area / inference speed” ratio, and does not require the inclusion of interfaces to connect with analog sensors or actuators [1-4]. On the other hand, the digital approach offers the advantages derived from using well-established design methodologies, which ease the programmability of the systems and their interconnection with conventional processing environments [5-7].

The implementation of low-cost high-speed digital fuzzy controllers is described in this paper. These characteristics are achieved by adopting some restrictions in the degree of overlapping of membership functions and by using simplified defuzzification methods. An important feature of this kind of controllers is its programmability: the fuzzy sets defining the antecedents and consequents, the rulebase, and the defuzzifier strategies can be selected according to a specific application domain.

In section II the basic aspects of fuzzy inference systems are analyzed and an efficient hardware architecture is presented. Section III shows the implementation details of two prototypes integrated in a 1.0 \( \mu \text{m} \) CMOS technology. The logical and temporal behavior of both circuits are described in section IV, while section V illustrates the use of these controllers as function approximators. Finally, some conclusions are summarized in section VI.

2. Controller Architecture

The rules describing fuzzy controller operation are expressed as linguistic variables represented by fuzzy sets. The controller output is obtained by applying an inference mechanism defined by the connectives used to link the rule antecedents, the implication function chosen, and the rule aggregation operator. In control applications, a defuzzifier stage is used to obtain a crisp value characterizing the output fuzzy set given by the inference process[8].

The most commonly used defuzzification methods, Center of Gravity (CoG) and Mean of Maxima (MoM), must sweep the whole universe of discourse to provide a solution. From a hardware viewpoint, this implies the use of massively parallel architectures (which means high area realizations) or sequential techniques (which imply slow system operation). A considerable reduction in both the inference time and the area of the fuzzy controller can be achieved when using simplified defuzzification methods, where the information provided by the rule consequents is codified by means of crisp parameters. The overall action of the ruleset is obtained by calculating the average of the different conclusions weighted by their grades of activation:

\[
\hat{y} = \sum_{i=1}^{r} \alpha_i \cdot w_i \cdot c_i / \sum_{i=1}^{r} \alpha_i \cdot w_i
\]  

The choice of \( w_i \) in (1) gives rise to different defuzzification methods. If \( w_i = 1 \), we obtain the so called Fuzzy Mean method (FM), its main drawback is that it does not consider the area and support of the fuzzy sets. If each \( w_i \) represents the area of a consequent fuzzy set the result is the Center of Sums method (CoS). In Weighted Fuzzy Mean (WFM) weight parameters proportional to consequent supports are used. Conversely, the Quality Method (QM) uses...
parameters inversely proportional to consequent supports to give more importance to crisper, rather than fuzzier, consequents [9].

Expression (1) can be implemented using simple arithmetic blocks. In addition, the summations in (1) are extended to the number of rules. Taking into account that only the active rules (those with $\alpha_i \neq 0$) will contribute to the solution, the inference time can be drastically reduced if we impose a limitation on the degree of overlapping of the antecedent membership functions [10] [11] and use an active rule driven mechanism to process the rules [12] [13].

An efficient hardware realization of fuzzy controllers based on the above two concepts was introduced by the authors in [14] and a generalization was presented in [15]. The controller architecture is depicted in Fig. 1. The membership function circuits (MFC) identify the active antecedents for each input value, and provides as many pairs (label, activation level) as degree of overlapping has been fixed for the system. The inference process is carried out by processing sequentially the active rules by means of an active-rule selection circuit composed by a counter-controlled multiplexer array. In each clock cycle, the membership degrees of a rule are combined through the MIN operator to calculate the activation level of the rule, while the antecedent labels address the position in the rule memory containing the parameters which define their corresponding consequent. Finally, a defuzzifier block performs the operations in (1).

Different circuit level solutions can be adopted to implement the building blocks in Fig 1. Membership function circuits can be built resorting to either vectorial or arithmetic approaches. Memory-based MFCs allow to define unrestricted fuzzy set shapes. However, when low cost in terms of silicon area is a primary target, an arithmetic block may provide better results. The rule memory can be implemented in RAM to improve the controller programmability or as a ROM or a combinational circuit to reduce silicon area. Finally, a defuzzifier block composed by two multipliers, a divider, and two adders implements all the defuzzification methods in (1) by adequately selecting the $w_i$ parameters stored into the rule memory.

3. Circuit Implementation

Based on the architecture described in section II, two fuzzy controller prototypes have been implemented. Both of them consist in three-input one-output controllers which can use up to 8 membership functions with an overlapping degree of 2 and a resolution of 6 bits. The whole rulebase memory is implemented in RAM (512x6 bits), thus allowing to define a different consequent for each possible rule. The defuzzifier block has been optimized for the Fuzzy Mean method (the first multiplier in Fig. 1 has been eliminated).

The first prototype uses a memory based MFC. Fig. 2-a shows a set of membership functions with maximum overlapping degree of two (triangular functions have been drawn for simplicity). The codification scheme is illustrated in Fig. 2-b. A memory of $E$ words of $(2N+\log_2 F)$ bits is required to implement the MFC, where $E$ is the number of elements in the input universe of discourse, $F$ is the number of antecedents used in the rules, and $N$ is the number of bits used to store the antecedents activation levels. The term $\log_2 F$ is required to store the first non-zero label. The code for the second label is obtained by incrementing by one the first code. According to these data, the size of the memories used for the MFCs of each input in this prototype is (64 x 15 bits).

The block diagram of the arithmetic MFC used in the se-
cond prototype is shown in Fig. 2-d. Fig. 2-c shows the fuzzy partition generated by this circuit. Breakpoints and slopes defining membership functions are stored in register banks. A counter driven by the master clock generates the register addresses. The breakpoint value is subtracted from the input and the result is multiplied by the slope. If \( x > x_i \), the product is stored in a latch and the counter is increased in the next active clock. The process continues until \( x < x_i \).

Both prototypes have been implemented in a 1.0 \( \mu \text{m} \) CMOS technology. Their microphotographies are shown in Fig. 3. The total area (including pads) of the prototype using memory-based MFCs is 14.36 mm\(^2\). A great deal of this area is consumed by the three MFCs located at the bottom part of the chip (Fig. 3-a). The prototype with arithmetic MFCs only requires one small memory to store the up to eight membership function parameters. The remainder circuits to generate the MFCs are implemented as standard cells, thus allowing a reduction in the chip area (12.53 mm\(^2\) in this case, Fig. 3-b).

4. Experimental Results

The circuits exhibit two operation modes: “load mode” and “inference mode”. In the “load mode” the knowledge

Figure 2. Memory-based MFC: a) fuzzy sets, b) memory organization. Arithmetic MFC: c) fuzzy sets, d) block diagram.

Figure 3. Fuzzy controllers with memory-based MFCs (a), and arithmetic MFCs (b).
base is stored in the controller through a load bus. An 8 bit bus has been chosen for compatibility with EPROMs and conventional microcontrollers. An internal control circuit generates the addresses for the different memory blocks and the write enable and load enable signals. In the “inference mode”, the controller output is evaluated according to its inputs. Two handshake signals provide information to the external interface. One signals the acquisition of an input data and another notifies the generation of a new output data.

A feature of these controllers is the use of pipelining to increase the system throughput. Based on this pipeline strategy, the number of clock cycles needed to perform an inference is the maximum among the number of antecedent labels (only when arithmetic MFCs are included), the number of active rules, and the number of output bits, plus one extra clock cycle. The circuit with memory-based MFCs obtains the antecedent activation levels in only one clock cycle. However, the circuit with arithmetic MFCs needs as many clock cycles to perform its operation as fuzzy sets are defined. One clock cycle is required by the active-rule selection mechanism to evaluate each active rule. Finally, using a sequential division circuit with shift-subtract/add nonrestoring technique, the division at the defuzzifier stage needs a clock cycle for each bit of the quotient. An extra clock cycle is introduced for transporting the data between the pipeline stages and to clear the registers. The number of active rules is the limiting factor for the two prototypes considered.

Fig. 4 shows simulation results for the first prototype. Data are sampled with the rise edge of signal “e_dato_val”. The circuit generates an output data after two pipeline stages. For this prototype pipelining requires 9 clock cycles. The circuit maximum clock frequency measured at laboratory was 27 Mhz, providing an inference speed over 3 MFLIPS.
The second prototype has three pipeline stages, because it was designed with the “arithmetic MFC” option. Fig. 5 shows experimental results. The data are sampled with “e_dato_val” signal and produce an output after three pipeline stages. As in the first prototype, the pipelining requires 9 clock cycles. The circuit operates at a 25 Mhz clock frequency, providing an inference speed of 2.77 MFLIPS.

5. Function Approximation

In order to prove the functionality of the chips their application to a fuzzy function approximation problem is described in this section. Fig. 6-a and Fig. 7-a show the graphical representation of the 2 input-1 output chosen functions [16]:

\[ F_1 = \frac{1}{1 + e^{10(x-y)}} \]  

(2)

\[ F_2 = \frac{1}{2} [1 + \sin(2\pi x)\cos(2\pi y)] \]  

(3)

Rule identification and membership functions adjust were obtained using a backpropagation learning algorithm. Seven membership functions were chosen to cover the universe of discourse of input variables and five fuzzy singletons for the consequents. The rulebase and the membership functions are shown in Fig. 6-b and Fig. 6-c for function F1 and in Fig. 7-b and Fig. 7-c for function F2.

Fig. 6-d illustrates the surface of function F1 generated by the controller with arithmetic MFCs. The corresponding error surface is displayed in Fig 6-e. The measured RMSE is 0.023, and the maximum error is 7.9 %.

Fig. 7 shows similar results for function F2. In this case, the RMSE is 0.033, and the maximum error is 7.9 %.

6. Conclusions

An efficient hardware realization of fuzzy controllers can be obtained by imposing some restriction in the shape of membership functions and using active rule driven inference methods. Two VLSI integrated circuits implementing fuzzy controllers based in these ideas have been presented in this paper. Even though both circuits use a simplified defuzzification method, their functionality has been proved by applying them to function approximation problems.
References


