OPTIMIZATION OF ADAPTIVE FUZZY PROCESSOR DESIGN

I. Baturone, S. Sánchez Solano, A. Barriga, J. L. Huertas.

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

XIII Conference on Design of Circuits and Integrated Systems (DCIS'98), pp. 316-321, Madrid, Noviembre 17-20. 1998

Optimization of adaptive fuzzy processor design

I. Baturone, S. Sánchez Solano, A. Barriga, J. L. Huertas.

Instituto de Microelectrónica de Sevilla, I.M.S.E.-C.N.M.
Avda. Reina Mercedes s/n. Edificio CICA. 41012-Sevilla.
Phone: 95 423 99 23 FAX: 95 423 18 32
E-mail: lumi@imse.cnm.es

Abstract

A fuzzy processor is programmed to provide an optimum output for solving a given problem. It could theoretically solve any problem (from a static point of view) if it is an universal approximator. This paper addresses the design of fuzzy processors aiming at a twofold objective: efficient adaptive approximation of different and even dynamically changing surfaces and hardware simplicity. Adequate programmable parameters and a fully-parallel architecture are selected. Mixed-signal blocks based on digitally programmed current mirrors are employed. Error-descent learning algorithms for tuning are discussed. Adaptive behavior is illustrated with an application to the on-line identification of a nonlinear plant.

I. Introduction

Fuzzy systems have drawn a great attention for their capability of translating expert knowledge expressed by linguistic rules into a mathematical framework. This is very interesting because as processes become more complex (non-linear and/or changing over time), the ability to describe them mathematically decreases and all that can be available is a linguistic description.

A typical multi-input single-output fuzzy system contains a set of IF-THEN rules like the following:

Rule i: IF x_1 is A_1^i and ... and x_u is A_u^i THEN y is B^i

where x_j (j=1,..., u) are the input and y is the output variables while A_j^i (i=1, ..., R) and B^i are, respectively, the antecedents' and consequent's fuzzy sets that represent linguistic values like "very big", "small", etc.

The inference is accomplished after three stages. The first one is fuzzification or calculation of the membership degrees, μ_j^i, of x_j to A_j^i. The second stage is rule processing, which is performed by (a) computing each rule's activation degree, $h_i = \wedge_j \mu_j^i$ (where $\wedge$ is a T-norm operator like the minimum, product, etc.), (b) by calculating each rule's conclusion, $y'_i = h_i \wedge B^i$, and (c) by computing its aggregation, $y' = \vee_i y'_i$ (where $\vee$ is a T-conorm operator like the maximum, sum, etc.). The last stage is defuzzification, which is aimed at obtaining a crisp output (by implementing a mean-of-maximum operator, a centroid, etc.) [1]. The crisp output provided by the fuzzy system not only depends on its rule base and its antecedents and consequents but also on the different operators employed to implement the T-norms, the T-conorm, and the defuzzification. To make an efficient selection of these operators, the designer should consider towards which applications the fuzzy system is addressed and how it is going to be implemented.

Fuzzy systems have been widely implemented with software on standard digital processors. However, when real-time operation and/or low area and power consumption are required the adequate solution is to implement them with dedicated hardware. An application specific fuzzy integrated circuit is the optimum solution for a particular problem. On the other side, a general-purpose fuzzy chip or fuzzy processor is better when a wide range of applications is considered. Among the applications of fuzzy systems there are fields like control and identification of nonlinear dynamical processes, nonlinear channel equalization, adaptive noise cancellation, and generation of linearizing or conditioning functions for nonideal sensors or signal pre-distortion [2-3]. While an application specific fuzzy IC provides a fixed output surface, a fuzzy processor should provide an optimum surface for different problems. Hence, the value of a fuzzy processor increases if it is a universal approximator of surfaces.

This paper addresses the design of fuzzy processors aiming at a twofold objective: efficient approximation of different and even dynamically changing surfaces and hardware simplicity. We will focus on implementing fuzzy inference engines with fixed T-norm, T-conorm and defuzzification operators suitable for these purposes. Approximation of a given output surface is achieved by programming adequate

parameters that define the antecedents and consequents of the rules. This is discussed in Section II. Another relevant point is to choose an efficient architecture. The selected architecture is briefly described in Section III. Digital programmability is preferred to ease user interface and to store the different parameters. This also eases manual or automatic tuning of the fuzzy processor when adaptation is required. Mixed-signal current-mode blocks are employed to implement rule processing. They admit digital programmability and provide the fuzzy processor with an analog I/O interface that allows direct communication with the usually analog sensors and actuators. These blocks are described in Section IV. Section V discusses adequate learning algorithms to tune the described fuzzy processor. A mixed-signal design is efficient because no high resolution is required in most of fuzzy applications. This is illustrated in Section VI with an example of application to the on-line identification of a non-linear plant.

II. A universal approximator fuzzy processor

Many types of fuzzy systems have been proved to be universal approximators [4-5] so that given a particular problem, they are theoretically effective. The problem is to select the most appropriate one.

From a hardware point of view, it is convenient to implement singleton fuzzy systems, also known as zero-order Takagi-Sugeno's systems. They represent the consequents by singleton values, c_i, and provide the following output:

$$z = \sum_{i=1}^{R} h_i \cdot c_i / \sum_{i=1}^{R} h_i \qquad (1)$$

Regarding representation of antecedents' fuzzy sets, piece-wise linear membership functions are easy to implement (especially working in current-mode) as well as very suited to be digitally programmed [6]. Another choice to do is the selection of the operator that combines the antecedents of a rule (to calculate h_i). The most popular ones are the minimum and the product operators. A multi-input minimum circuit is simple to design in current-mode while a multiplier is more complex [6]. However, the output surface provided by a fuzzy system that employs the minimum operator is generally non-linear while employing the product operator it is piece-wise multi-linear or multi-affine. The authors in [7] demonstrate that fuzzy systems whose input membership functions cover the input universes of discourses as shown in Figure 1 have better approximation accuracy when employing the product instead of the minimum connective. Since one of our objectives it to achieve efficient approximation, we will focus on implementing these fuzzy systems with the product operator. They provide the following output:

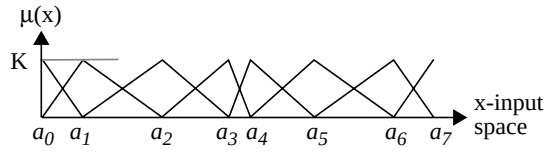


Fig. 1: Choice of the covering of the input spaces.

$$z = \sum_{i=1}^{R} h_i \cdot c_i / K \quad \text{with} \quad h_i = \prod_{j=1}^{u} \mu_j^i \qquad (2)$$

where K is a constant ($\Sigma_i h_i = K$).

The programmable parameters are the points a_i of each input variable and the singletons of the rules. We describe the architecture and the building blocks of these processors in the following.

III. A fully-parallel architecture

The building blocks required to implement the selected fuzzy system are: (a) MFC's that implement antecedents' membership functions, (b) T circuits to compute h_i by antecedents' connection (by product operator), and (c) CONS blocks that implement the product $h_i \bullet c_i$. A final divider circuit is not required because the sum $\Sigma_i h_i$ is constant.

Since a singleton fuzzy system is inherently parallel in its input variables and rules, there is always a trade-off between high inference speed (parallel processing) and low silicon area (sequential processing).

From the inherent grid partition of the selected type of fuzzy systems, an architecture that shares the MFC's is preferred. A relevant feature of fuzzy systems is that input variables typically trigger a small number of the total number of rules, so that the maximum number of rules that are simultaneously active is α^u, where α is the overlap factor, that is, the maximum number of active fuzzy sets per input. In our case, α is 2. Fully-digital fuzzy chips have been reported in the literature that take advantage of this feature, thus processing only the α^u active rules [8-9]. To allow parallel processing of these rules, the proposal in [8] is to employ α^u copies of the rule memory and to use multibit computing operators, which are very area consuming. On the other side, the digitally-programmable analog fuzzy chips reported in the literature offer parallel computing with lower hardware resources but they implement much less rules than the fully-digital ones (for instance 4 [10], 9 [11], 15 [12], or 49 [13] against 102 [14], 128 [15], 512 [16], or 960 [9]). In addition, these digitally-programmable analog fuzzy chips do not optimize digital part because the programmable parameters are stored in digital registers and the selection circuitry consists in extensive matrixes of switches (multi-port-like digital

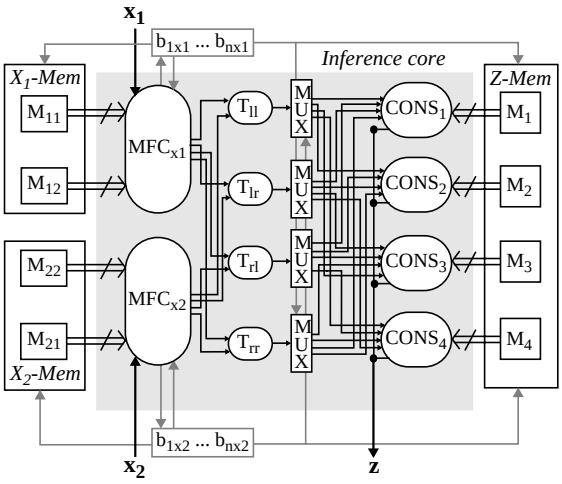


Fig. 2: Selected architecture [17].

memories) which occupy a large area (for instance, the 93% of the total area of the programmable chip described in [12] is occupied by the digital part).

To avoid these problems we have selected the architecture presented in [17] that shares the MFC's but not the CONS's and that implements all the possible rules, L^u, to be suitable for many applications (where L is the number of fuzzy sets per input). This architecture allows optimization of both the analog and digital part of a fully-parallel fuzzy processor. The analog core is optimized by using an active-rule driven scheme. This scheme reduces not only the number of MFC's required, from $L \cdot u$ to $2 \cdot u$, but also the number of T's and CONS's, from L^u to 2^u. The analog core mainly depends on u, and basically remains the same independently of the number of labels, L, per input, and the total number of rules, L^u (for example, it is basically the same for chips with 3x3=9 rules or 13x13=169 rules). The digital part is also optimized by using an adequate memory organization that makes it possible to retrieve all the required parameters in parallel from standard digital RAM's without a need for replication or multi-port costly memories. Figure 2 illustrates this architecture for the fuzzy system here selected in the case of two input variables. Figure 3 shows the partition of the antecedents' and

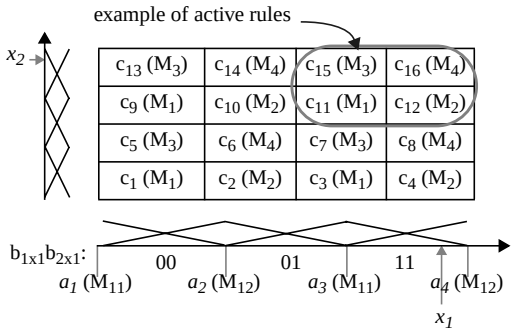


Fig. 3: Rule table that illustrates the partitions of the antecedents' and consequents' memories.

consequents' memories for the case of four labels per input. In this example, the odd a_i points of the input variables are stored in the M_1 parts of the corresponding input memory while the even a_i points are stored in the M_2 parts. Regarding the consequents, the M_1 part of the Z-Mem, for instance, stores the singletons c_1, c_3, c_9, and c_{11}. Considering adaptive fuzzy processors, this architecture is also advantageous since, given an input, only the parameters associated with the active rules take part in the adaptive phase, similarly to that happens in the inference phase.

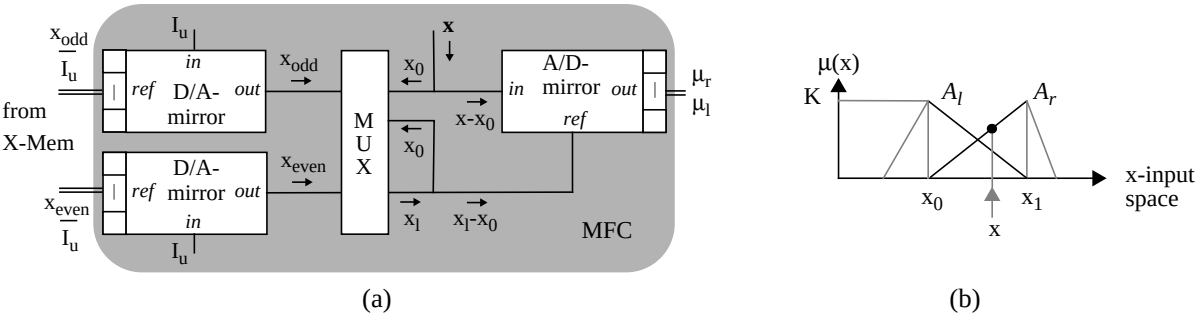
IV. Building blocks of the inference core

The MFC's have to generate the following expression (according to Figure 4b):

$$\frac{K(x-x_0)}{x_1-x_0} \tag{3}$$

where x is the usually analog input and x_1, x_0 are the digitally programmable parameters (a_i points) given by the X-Memory.

Two substractions and a division have to be implemented. If digital circuitry is employed, one A/D converter is required to convert each input variable to the digital domain. Our proposal is to employ mixed-signal processing so that an A/D is exploited to implement the division (see Figure 4a). Since no high resolution is required by most of fuzzy applications (we will see an example in Section VI), this A/D can be designed as a continuous-time algorithmic con-



(a) (b)

Fig. 4: (a) Scheme of the MFC. (b) The two active labels per input.

verter based on current mirrors (That is why it is named A/D-mirror in Figure 4a). The rest of the MFC consists of two digitally programmed current mirrors, named D/A-mirror in Figure 4a. They enable to implement the substraction $x\text{-}x_0$ and $x_1\text{-}x_0$ by wire connection (supposing that the input variable is represented by a current). Description of these A/D and D/A-mirrors, that is the selection of their structures and data about their silicon area occupation, power consumption and operation speed can be found in [18]. As an example, the combination of a 5-bit A/D- and a D/A-mirror occupy an active area of 0.08 mm^2 in a 2.4-μm CMOS technology. From Hspice simulations, the response time of the combination is 120 ns for a static power consumption of 409 μW (working at a 3-V power supply).

The outputs of an MFC so designed are two digital words, as follows:

$$\mu_r(x) = Q\left(\frac{x-x_0}{x_1-x_0}\right) \quad \text{and} \quad \mu_l(x) = \overline{\mu}_r(x) \tag{4}$$

where Q(.) is a quantization operator.

The T circuits implement a product connective by also employing digitally programmed current mirrors as shown in Figure 5a. The output currents of these circuits represent the activation degrees of the active rules. The CONS's are also digitally programmed current mirrors (Figure 5c). Their output currents are wired out to provide the global crisp output of the fuzzy processor.

A feature of an active-rule driven architecture is the need for additional circuitry to select the antecedents' parameters, a_i, and the consequents' parameters, c_i, from the digital memories. This additional circuitry compares each input variable with the points a_i of its corresponding space to obtain a control digital word of n bits, $(b_1, ..., b_n)$ in Figure 3, n being the integer bigger than or equal to $\log_2(L\text{-}1)$. The comparison can be done in parallel, using L-2 current comparators and programmable current mirrors, or in series, following a binary-tree scheme. In the last case, the operation takes n clock phases. The n-bit word controls which a_i point is x_0 and x_1 (this is the purpose of the MUX blocks within the MFC's) and which c_i is c_{rr}, c_{ll}, etc. (this the purpose of the MUX blocks between the T's and the CONS's). The MUX blocks are digital switches controlled by this word. Once the control digital word is obtained, the whole processing of all the active rules is carried out in parallel, in a few hundreds of nanoseconds (depending on the technology).

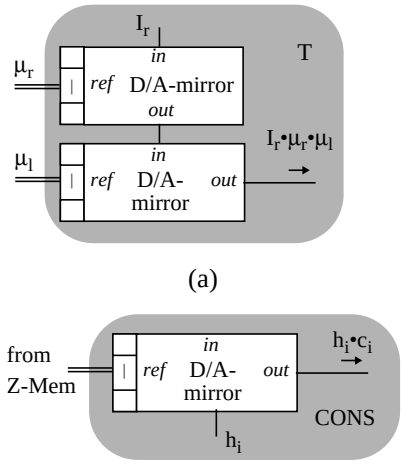


Fig. 5: (a) T and (b) CONS circuits used.

V. Adequate learning algorithms

A fuzzy processor designed with the described architecture and building blocks approximate a given output surface by properly programming the values of the a_i points and of the singletons. If the fuzzy processor works in a dynamically changing environment, it has to be combined with a tuning/learning block that updates the programmable parameters to match the new situation. We briefly describe in the following simple learning methods that can be implemented off-chip or on-chip with the fuzzy processor. We will focus on supervised learning methods, which can be employed when a set of training patterns is available.

Tuning of singleton values

Since the output of a singleton fuzzy system is a linear function of the singleton values, a gradient-descent method is very suitable to optimize them. The generic gradient-descent updating equation for a singleton value, c_i, is:

$$c_i\big|_{new} = c_i\big|_{old} - \eta \cdot \frac{\partial E}{\partial c_i} \tag{5}$$

where η, the learning rate, is a suitably chosen constant and E, the error, is a performance index of how well the desired function, g, is approximated by the fuzzy system output, z. E is usually defined as the mean quadratic error over a finite set of training data or a finite time interval [k+1-T, k]:

$$E = \frac{1}{T} \cdot \sum_{p=k+1-T}^{k} (z-g)^2\Big|_p \tag{6}$$

Taken for z the expression in (2), the parameter c_i is then adjusted as:

$$c_i(k+1) = c_i(k+1-T) - \frac{2\eta}{T} \sum_{p=k+1-T}^{k} (z-g)\,\frac{h_i}{K}\Big|_p \tag{7}$$

This updating can be performed off- or on-chip. The circuitry required for on-chip implementation is described in [19]. Equation (7) represents a block or batch learning algorithm where each singleton value

is updated after the presentation of T patterns. More adequate for on-chip implementation, although less effective in general, is on-line learning, where the singleton value is updated after the presentation of 1 pattern (T=1). For certain applications like adaptive noise cancellation batch learning is required. For other applications, like identification of non-linear dynamical plants, on-line learning can be enough.

Tuning of antecedents' values

In some cases, the input spaces can be uniformly partitioned and only the singleton values can be adjusted. However, there are several applications for which this solution would conduct to very fine partitions and consequently many rules to achieve a given approximation accuracy. In these cases, adjusting of antecedents parameters is also convenient, although this is much more difficult than consequent tuning. One of the causes is that the dependence of the fuzzy system's output on the antecedents' parameters is nonlinear so that the optimization process based on gradient-descent techniques can be trapped at local minima. The tuning parameters are the points, a_i, of the input spaces (remember Figure 1). If L labels cover an input space, we will have L-2 tuning parameters in that space because the extreme points are fixed.

Another cause that makes it difficult antecedent tuning is that the expressions of the partial derivatives of the error function are much more complex than for the consequents. To avoid this problem, a weight-perturbation learning algorithm has been chosen [20]. This is an error-descent method with a very simple updating equation:

$$a_i(k+1) = a_i(k+1-T) - \beta \cdot \sum_{p=k+1-T}^{k} \Delta E|_p \qquad (8)$$

where β is a suitably chosen constant and $\Delta E|_p$ is the error variation for the pattern p and for a small perturbation in the parameter a_i. $\Delta E|_p$ is calculated as:

$$\Delta E|_p = |z_{pert} - g| - |z - g| \qquad (9)$$

where z_{pert} is the output of the fuzzy system when parameter a_i is perturbed. Signal $\beta \cdot (z-g)$ is supposed to be provided from outside the chip, as in the consequents and equation (8) can be implemented off- or on-chip [19].

VI. Example of application: on-line identification of a non-linear plant

Adaptive fuzzy chips of low resolution (below 8 bits) and with a few parameters to adjust can be succesfully employed in several applications. To illustrate this, we have applied the above described learning algorithms to tune a fuzzy processor to identify the following nonlinear process (Figure 6a):

$$f(x,y) = \frac{\sin x}{x} \cdot \frac{\sin y}{y} \qquad \text{with } x,y \in [-1,1] \qquad (10)$$

An adaptive fuzzy processor with 8 labels per input (6+6+64=76 tuning parameters) has been described by C language. A resolution of 7 bits has been considered for the D/A and D/A-mirrors of the building blocks. 121 training patterns have been used. When only the 64 singleton values are on-line tuned, the final root-mean squared error (RMSE) reached after 9 epochs is 3.4% (Figure 6b). If the antecedents'

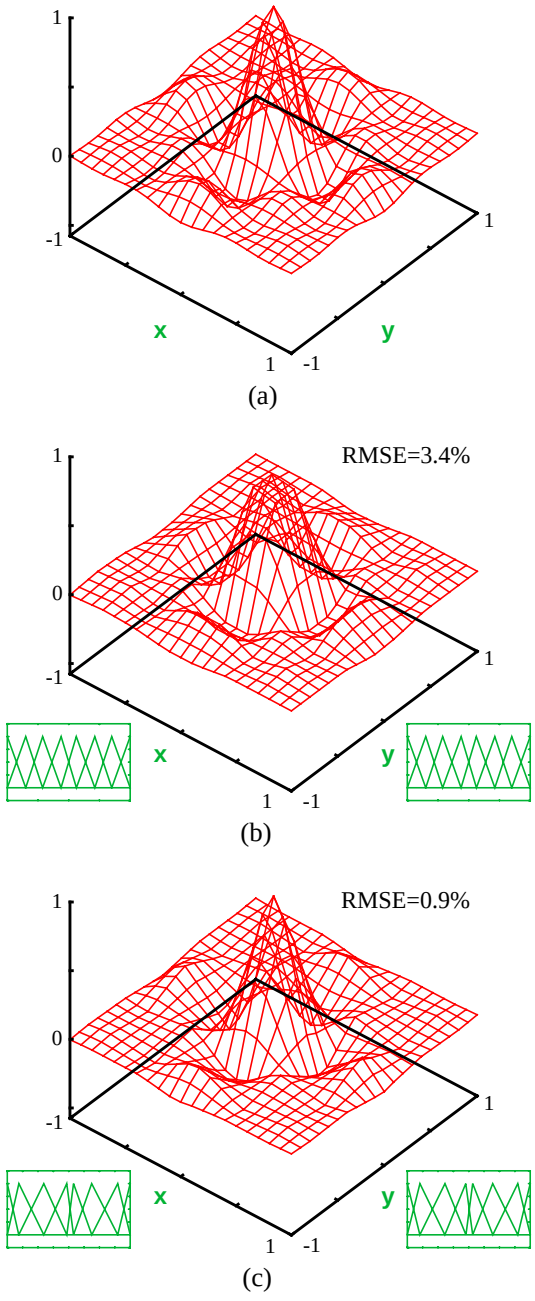


Fig. 6: On-line identification of a nonlinear process: (a) Desired surface. Approximation obtained: (b) with only consequent tuning and (c) when antecedents and consequents are tuned.

parameters (12 points) are also tuned on line, the RMSE decreases to 0.9% after 85 epochs (Figure 6c). Initially, the antecedents' membership functions covered uniformly the input spaces and all the singleton values were 0.5.

Robustness of an adaptive fuzzy processor against errors in its circuitry has been confirmed in this example. A 20% gain error was introduced in one of the D/A-mirrors of a T circuit and the same training was performed. An RMSE of 1.0% was obtained after 170 epochs.

VII. Conclusions.

This paper has focused on the design of adaptive fuzzy processors. Optimization of this design has been done taking into account theoretical and practical issues. Considering theoretical issues, we have decided to implement zero-order Takagi-Sugeno's systems that employ the product as the connective operator of the antecedents and that cover the input spaces with triangle-shaped membership functions whose overlapping is 50%. These systems offer good approximation and tuning properties. Considering hardware implementation, we have employed a fully-parallel active-rule driven architecture, where the programm-able parameters are stored in standard digital RAM's and the computing blocks are implemented with mixed-signal circuits based on current mirrors. Error-descent learning algorithms have been applied to tune this processor. A mixed-signal design is efficient because high resolution is not required in many applications as shown with an example of a plant identification.

VIII. REFERENCES

[1] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller", Parts I y II, IEEE Trans. Syst., Man and Cybern., vol. 20, no. 2, pp. 404-432, 1990.

[2] L. X. Wang, J. M. Mendel, "Back-propagation fuzzy systems as nonlinear system identifiers", Proc. Int. Conf. on Fuzzy Systems, San Diego 1992, pp. 1409-1418.

[3] C.-T. Lin and C.-F. Juang, "An adaptive neural fuzzy filter and its applications", IEEE Trans. Syst., Man, and Cybern., vol. 27 (4), pp. 635-656, August 1997.

[4] L-X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning", IEEE Trans. Neural Networks, vol. 3, no. 5, pp. 807-814, Sept. 1992.

[5] J. L. Castro, "Fuzzy logic controllers are universal approximators", IEEE Trans. Syst., Man, and Cybern., vol. 25, no. 4, pp. 629-635, April 1995.

[6] I. Baturone, S. Sánchez-Solano, A. Barriga, J. L. Huertas, "Implementation of CMOS Fuzzy Controllers as Mixed-Signal IC's", IEEE Trans. on Fuzzy Systems, vol. 5, no. 1, pp. 1-19, February 1997.

[7] X.-J. Zeng, M. G. Singh, "Approximation accuracy analysis of fuzzy systems as function approximators", IEEE Trans. on Fuzzy Systems, vol. 4 (1), pp. 44-63, Feb. 1996.

[8] Tzi-cker Chiueh, "Optimization of fuzzy logic inference architecture", IEEE Computer, vol. 24, no. 5, pp. 67-71, May 1992.

[9] M. Sasaki, F. Ueno, T. Inoue, "An 8-bit resolution 140 KFLIPS fuzzy microprocessor", Fifth IFSA World Congress, pp. 921-924, Seul 1993.

[10] T. Miki, H. Matsumoto, K. Ohto, T. Yamakawa, "Silicon implementation for a novel high-speed fuzzy inference engine: mega-flips analog fuzzy processor", Journ. of Intelligent and Fuzzy Systems, vol. 1, no. 1, pp. 27-42, 1993.

[11] I. Baturone, S. Sánchez-Solano, A. Barriga, J. L. Huertas, "Flexible fuzzy controllers using mixed-signal current-mode techniques", Proc. FUZZ-IEEE'97, pp. 875-880, Barcelona 1997.

[12] N. Manaresi, E. Franchi, R. Guerrieri, G. Baccarani, "A field programmable analog fuzzy processor with enhanced temperature performance", Proc. ESSCIRC'96, pp. 152-155, Neuchatel 1996.

[13] J. Oehm, M. Grafe, T. Kettner, K. Schumacher, "Universal low cost controller for electric motors with programmable characteristic curves", IEEE Jour. Solid-State Circuits, vol. 31, no. 7, pp. 1041-1045, July 1996.

[14] H. Watanabe, W. Dettloff, K. E. Yount, "A VLSI fuzzy logic controller with reconfigurable, cascadable architecture", IEEE Jour. Solid-State Circuits, vol. 25, no. 2, pp.376-382, April 1990.

[15] T. Katashiro, "A fuzzy microprocessor for real-time control applications", Fifth IFSA World Congress, Seúl 1993, pp. 1394-1397.

[16] C. J. Jiménez, S. Sánchez Solano, A. Barriga, "Hardware implementación of a general purpose fuzzy controller", Proc. IFSA World Congress, Sao Paulo 1995, pp. 185-188.

[17] I. Baturone, A. Barriga, S. Sánchez Solano, J. L. Huertas, "Mixed-signal design of a fully parallel fuzzy processor", Electronics Letters, vol. 34, no. 5, pp. 437-438, March 1998.

[18] I. Baturone, S. Sánchez Solano, J. L. Huertas, "A current-mode A/D-D/A memory and computing block", Proc. of this conference, DCIS'98.

[19] I. Baturone, S. Sánchez Solano, J. L. Huertas, "Mixed-signal VLSI design of adaptive fuzzy systems", Proc. 7th IEEE Int. Conf. on Fuzzy Systems, Anchorage, May 1998, pp. 25-30.

[20] M. Jabri, B. Flower, "Weight perturbation: an optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks", IEEE Trans. on Neural Networks, vol. 3 (1), pp. 154-157, Jan. 1992.