MIXED-SIGNAL VLSI DESIGN OF ADAPTIVE FUZZY SYSTEMS

I. Baturone, S. Sánchez-Solano, J. L. Huertas

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

Seventh IEEE International Conference on Fuzzy Systems (FUZZ-IEEE’98),
Abstract

Hardware realization of adaptive fuzzy systems is discussed in this paper. A highly-parallel architecture based on an active-rule driven scheme is considered so that only the parameters associated with the active rules take part in the inference and learning phases. Mixed-signal circuits are used to implement a gradient-descent algorithm for tuning consequents’ values and a weight perturbation method to adjust suitably chosen antecedents’ parameters. Several applications of adaptive fuzzy chips are discussed showing that low-cost realizations are feasible.

1. 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. Another advantage is that fuzzy systems as simple as zero-order Sugeno’s type or singleton fuzzy systems have been proved to be universal approximators and, hence, potentially capable of describing any process [1].

As a consequence, fuzzy systems can be applied to many 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 [2-4].

The global structure of a singleton fuzzy system is defined by a rule base which connects the antecedents’ fuzzy sets that cover the input spaces with the consequents’ singleton values of the output space. The detailed structure is defined by the parameters that represent the antecedents’ membership functions and the consequents’ singleton values. A current research area is the development of adaptive fuzzy systems which can automatically find their structure using linguistic information and learning algorithms, most of them taken from the neural network domain. Non-supervised algorithms have been employed to define the global structure [4] and supervised ones for the detailed structure [2-4].

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 (fuzzy chips). In particular, realization of adaptive fuzzy chips or fuzzy hardware with on-chip learning is a promising research area because they can work autonomously into embedded systems and efficiently against parameter perturbation of the process or the fuzzy hardware itself.

Several issues concerning hardware realization of adaptive fuzzy systems are addressed in this paper. The selected architecture is described in Section 2, while the circuitry required to implement the inference and learning methods is discussed in Section 3. Illustrative examples of applications to adaptive noise cancellation and on-line identification of nonlinear processes are included.

2. Selection of an efficient architecture

An adaptive fuzzy chip should contain circuitry to implement the fuzzy inference method and to carry out the learning algorithm. Selection of an efficient architecture is crucial to simplify total circuitry and to provide a high inference and learning speed.

Singleton fuzzy systems infer a crisp output, \( z \), by implementing the following formula:

\[
  z = \frac{\sum_{r=1}^{R} h_r \cdot c_r}{\sum_{r=1}^{R} h_r}
\]  

(1)

where \( R \) is the number of rules, \( c_r \) is the consequent’s singleton value of the \( r \)-th rule, and \( h_r \) is its activation degree (usually calculated as the minimum or product of the input variables’ membership degrees to their corresponding fuzzy sets).

The building blocks required are: (a) MFC’s that implement antecedents’ membership functions, (b) \( T \) circuits to
compute \( h_r \) by antecedents’ connection, (c) CONS blocks that implement the product \( h_r \cdot c_r \), and (d) a DIV circuit to calculate the final division \( \Sigma h_r \cdot c_r / \Sigma h_r \).

An architecture where each rule has its own MFC’s, T and CONS circuits (a data path for each rule) offers the biggest flexibility. However, it is the most costly from a hardware point of view because several MFC’s and CONS blocks have to be usually repeated in different rules. An adaptive fuzzy chip with \( u \) input variables would require \( R^u \) MFC’s and \( R \) CONS blocks with their corresponding memory and learning circuits to adapt and to store the parameters that define them.

Sharing MFC’s and CONS circuits among the rules is the most efficient solution for application-specific fuzzy chips. If \( L \) is the number of fuzzy sets per input variable (taken equal for all the inputs for simplicity) and \( M \) is the number of singleton values, the number of MFC’s and CONS blocks is reduced to \( L^u \) and \( M^u \) respectively. When adaptive chips are considered, the complication of this approach is that memory and learning circuitry is required to define not only the antecedents’ and consequents’ parameters (detailed structure) but also the rule base (global structure). Hence, non-supervised learning algorithms like clustering techniques and competitive learning or ART-based algorithms [4] should be implemented.

As a trade-off between flexibility and hardware complexity, we have selected an architecture that shares the MFC’s but not the CONS blocks (thus avoiding complex algorithms to learn the rule base) and that implements all the possible rules, \( L^u \) (to be suitable for many applications). A 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 \( \alpha^u \), where \( \alpha \) is the overlap factor, that is, the maximum number of active fuzzy sets per input. In practice, \( \alpha \) is usually 2. An active-rule driven scheme takes advantage of this feature and reduces not only the number of MFC’s required, from \( L^u \) to \( \alpha^u \), but also the number of T and CONS circuits, from \( L^u \) to \( \alpha^u \). Hence, we implement the selected architecture with a rule-driven approach. Novelties of the selected architecture (more widely described in [5]) is the employment of current-mode mixed-signal circuits to carry out the inference process (exploiting the low area of analog domain) and an efficient memory organization to allow parallel processing of both the rules and the operations within each rule. Figure 1a illustrates this architecture for the case of two input variables and a maximum overlapping degree of two. The MFC’s and CONS blocks contain a dashed part that represents the possible circuitry to adapt their parameters. Regarding adaptive chips, this architecture is advantageous in terms of memory and learning circuitry, as illustrated schematically in Figure 1b. Given an input, only the parameters associated with the active rules take part in the learning phase, similarly to that happens in the inference phase.

Figure 1: (a) Proposed architecture for the case of two input variables and a maximum overlapping degree of two. (b) Schematic comparison of several architectures.
3. Circuitry for learning

Tuning of the parameters that define the antecedents and consequents of a fuzzy system can be done with a reinforcement learning [6] or with supervised methods, if a set of training patterns is available. We will focus on the last case.

3.1. 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:

\[
\frac{c_i}{new} = c_i \bigg|_{old} - \eta \frac{\partial E}{\partial c_i} \tag{2}
\]

where \( \eta \), the learning rate, is a suitably chosen constant and \( E \), the error, is a performance index of how well the desired function, \( z_d \), 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 \([k+1-T, k]\):

\[
E = \frac{1}{T} \cdot \sum_{p=k+1-T}^{k} (z-p)^2_p \tag{3}
\]

Taken for \( z \) the expression in (1), the parameter \( c_i \) is then adjusted as:

\[
c_i(k+1) = c_i(k+1-T) - \frac{2\eta}{T} \sum_{p=k+1-T}^{k} (z-p) \frac{h_i}{\Sigma h_i} \tag{4}
\]

Equation (4) represents a block or batch learning algorithm where each singleton value is updated after the presentation of \( T \) patterns. More adequate for hardware 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, the desired output is contaminated by an additional signal that should be averaged out during the training process, so that batch learning is required. For other applications, like identification of non-linear dynamical plants, on-line learning can be enough. In the following, we describe the circuitry selected to implement equation (4) and its simplification when on-line learning is considered.

Given a training pattern \( p=k+T \), and a consequent parameter \( c_i \), the part of the consequent memory (Z-Mem) associated with \( c_i \) is divided in two areas that store the value of \( c_i \) before training, \( c_i(k+1-T) \), and the accumulated value \( c_i(k+1-T) \) defined as:

\[
c_i(k+1-T) = c_i(k+1-T) - \frac{2\eta}{T} \sum_{p=k+1-T}^{k} (z-p) \frac{h_i}{\Sigma h_i} \bigg|_{p=k+1-T} \tag{5}
\]

The quantization of the singleton values and specially the quantization of their increments influence the value of the final error obtained after training. The number of the quantization levels depends on the application. When high resolution is required, the calculation of the \( c_i \) is better carried out by analog integrators. Otherwise, they are discretized and stored in digital memories, as it is done with the final accumulated value \( c_i = c_i \bigg|_{new} \). In the following, we will describe the last approach (that will be used in one of the examples). If on-line learning is considered, circuitry is reduced because this additional analog or digital memory is not required.

During the inference phase (\( W=\text{"1"} \)), if rule \( i \) is active, the digital value \( c_i(k+1-T) \) is selected to weight the current \( h_i \) that represents the rule’s activation degree. This is carried out in the CONS block, which is a digitally-programmed current mirror (D/A-mirror). The contribution of all the active rules is wired-summed and conducted to the DIV block to calculate \( z \). Figure 2 shows with black lines the circuitry associated with a singleton value that is required during the inference process.

To carry out the learning process, the parameter change \( \zeta^*_i(z-p) \cdot h_i / \Sigma h_i \) has to be computed and the new value \( c_i \) has to be written in the digital consequent memory. As in the inference process, computing is performed in current-mode analog domain. The current that represents the new value, \( c_i \cdot \zeta^*_i(z-p) \), is discretized by the A/D mirror (an A/D converter based on current mirrors) during \( D=\text{"1"} \). Hence, the digital value \( c_i \) can be written when \( S=\text{"1"} \). The block A/D-D/A-mirror performs as an efficient analog memory, as de-

![Figure 2: Circuitry associated with the singleton value of an active rule.](image-url)
scribed in [7]. Once \( t = T \), the value is also stored in the area memory of the Z-Mem corresponding to \( c_i \).

The A/D-mirror is exploited to calculate the discrete value of \( h_i / \sum h_r \), \( Q(h_i / \sum h_r) \) during the learning phase. Since the values of \( z_d \) and \( \zeta \) depend on the particular problem and even \( z \) can be an external signal related with the fuzzy system output, signal \( \zeta \cdot (z - z_d) \) is obtained outside the chip with a suitable conditioning circuitry that allows continuous variation of \( \zeta \).

In the following, we illustrate two possible applications of adaptive fuzzy chips whose singleton values are adjusted as described above.

Example 1: On-line identification of nonlinear dynamical plants.

For this application, the flow diagram of the configuration is shown in Figure 3a. The non-linear plant considered (usually reported in the literature [2]) is governed by the following difference equation:

\[
y(k + 1) = 0.3y(k) + 0.6y(k - 1) + g[u(k)]
\]

(6)

where the unknown function, \( g(\cdot) \), has the form:

\[
g[u] = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u)
\]

(7)

The following series-parallel model is used to identify the plant:

\[
y_r(k + 1) = 0.3y(k) + 0.6y(k - 1) + F[u(k)]
\]

(8)

where \( F[u(k)] \) is the response of an adaptive fuzzy system.

This configuration has been simulated with C language considering a fuzzy system with 18 rules whose antecedents’ fuzzy sets (triangular shaped) cover the input space uniformly and whose singleton values are updated on-line after each time step, \( k \). During training, the input to the plant and to the model was a sinusoid \( u(k) = \sin(2\pi k/40) \) and initially, the 18 singleton values were zero. Considering a resolution of only 4 bits for the A/D-D/A-mirrors and the Z-Mem memory of the chip, that is, for the singleton values, the model follows the output of the plant as illustrated in Figure 3b. This figure shows the output of the identification model (grey line) and the plant (black line) for the input \( u(k) = \sin(2\pi k/250) \) for \( 0 \leq k \leq 250 \) and \( 501 \leq k \leq 700 \) and \( u(k) = 0.5\sin(2\pi k/250) + 0.5\sin(2\pi k/25) \) for \( 251 \leq k \leq 500 \), after the identification model was trained for 70 time steps.

Example 2: Adaptive noise cancellation system.

Application of an adaptive fuzzy system to noise cancellation problem is shown in Figure 4a. The information signal, \( s \), is corrupted by a noise, \( n_0 \), which is a nonlinear

\[
\begin{align*}
\text{Plant} & \quad \rightarrow \\
\text{Model} & \quad \rightarrow \\
\text{Fuzzy system} & \quad \rightarrow \\
\text{Reference} & \quad \rightarrow
\end{align*}
\]

Figure 3: (a) Identification of a dynamic plant. (b) Output of the model (grey line) and the plant (black line) after the training process.

\[
\begin{align*}
\text{Plant} & \quad \rightarrow \\
\text{Model} & \quad \rightarrow \\
\text{Fuzzy system} & \quad \rightarrow \\
\text{Reference} & \quad \rightarrow
\end{align*}
\]

Figure 4: (a) Adaptive noise cancellation. (b) The corrupted signal, \( x \). (c) The information signal, \( s \). (d) The recovered signal, \( x_r \).
function of a noise source \( n_f \) due to the existence of a nonlinear channel. In order to generate a replica of \( n_p \), the fuzzy system is trained with pairs of data \((n_f, s+n_p)\) with a batch learning. The example 4 reported in [4] has been considered, where:

\[
s(k) = \sin(0.06k) \cdot \cos(0.01k) \quad \text{and} \quad n_0(k) = 0.6 \cdot [n_1(k)]^3
\]

The configuration has been described by C language. The fuzzy system was considered to be implemented as a chip with 9 rules whose antecedents’ membership functions (triangular shaped) covered the input space uniformly. The 9 singleton values were updated after each epoch of 628 training data. The learning and memory circuitry of the chip is taken with 6 bits of resolution. This is enough for a good performance, as illustrated in Figure 4. Figure 4d shows the recovered signal, \( x_r \), after 16 epochs of training.

### 3.2. Tuning of antecedents’ parameters

In the former examples, the input spaces were uniformly partitioned and only the singleton values were 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. To reduce this problem we have focused on fuzzy systems that use product as antecedents’ connective and whose input fuzzy sets (represented by triangular membership functions) are normalized, that is, the overlapping degree is two singleton values. In this case, the MFC already contains the required D/A-mirrors and one A/D-mirror. They are provided as digital words of expressions:

\[
\mu_r = Q((x - x_l)/(x_r - x_l)) \quad \text{and} \quad \mu_f = \bar{\mu}_r
\]

Implementation of equation (10) is similar to that of singleton values. In this case, the MFC already contains the required D/A-mirror and the A/D-mirror to write the new antecedent’s parameter. As happens with consequents, online learning is more adequate for hardware implementation.

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

where \( \beta \) 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:

\[
\Delta E|_p = [z_{\text{pert}} - z_d] - [z - z_d]
\]

where \( z_{\text{pert}} \) is the output of the fuzzy system when parameter \( a_i \) is perturbed. Signal \( \beta(z - z_d) \) is supposed to be provided from outside the chip, as commented for the consequents.

In the former section, no constraint was imposed on the antecedents’ membership functions or the antecedents’ connective. In particular, in the examples previously described, the MFC’s were supposed to provide a continuous transfer function, like those proposed in [8]. If antecedent tuning is also performed, the MFC’s and T circuits are better realized with the circuitry shown in Figure 6. Given an input \( x \), the two membership degrees, \( \mu_r \) and \( \mu_f \), are calculated with two D/A-mirrors and one A/D-mirror. They are provided as digital words of expressions:

\[
\mu_r = Q((x - x_l)/(x_r - x_l)) \quad \text{and} \quad \mu_f = \bar{\mu}_r
\]

Figure 5: Points \( a_i \) are the tuning parameters of the antecedents.

Figure 6: (a) MFC’s and (b) T circuits used.
since no additional digital or analog memory is required. A difference with the consequent tuning is that antecedent perturbation is performed sequentially (instead of concurrently) over the antecedents’ parameters of the \( \alpha u \) input fuzzy sets that are active for each training pattern.

In the following, we illustrate antecedent tuning with an application to nonlinear process identification.

**Example 3: On-line identification of nonlinear processes.**

We have considered the following nonlinear process (Figure 7a):

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

(13)

An adaptive fuzzy system with 8 labels per input (6+6+64=76 tuning parameters) and 7-bit resolution has been described by C language. 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 7b). If the antecedents’ parameters (12 points) are also tuned on line, the RMSE decreases to 0.9% after 85 epochs (Figure 7c). Initially, the antecedents’ membership functions covered uniformly the input spaces and all the singleton values were 0.5.

Robustness of an adaptive fuzzy chip 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.

**Acknowledgment:** We gratefully acknowledge support from C.S.I.C. (Consejo Superior de Investigaciones Científicas) under a postdoctoral grant of the first author.

4. Conclusions

Selection of an efficient architecture allows simple circuitry to carry out the fuzzy inference and the learning algorithms. Digitally-programmed (A/D) current mirrors coupled with current-mode A/D blocks take advantage of analog domain to implement the computing required and of digital domain to store the parameters. On-line tuning of the parameters is more adequate for hardware realization than batch learning. In particular, the tuning of consequents is easily realized and usually requires a few epochs of training. Adaptive fuzzy chips of low resolution (below 8 bits) and with a few parameters to adjust can be successfully employed in several applications, as illustrated in this paper.

5. References


