Mathware & Soft Computing 8 (2001) 303-319

Microelectronic Design of Universal Fuzzy Controllers

I. Baturone and S. Sánchez-Solano
Instituto de Microelectrónica de Sevilla
Centro Nacional de Microelectrónica - CSIC
Avda. Reina Mercedes, s/n. Edif. CICA, 41012, Sevilla
e-mail: lumi@imse.cnm.es

Abstract

Fuzzy controllers have been proven to be universal, that is, they can provide any control surface. Their microelectronic implementation is very suitable to achieve high-speed (real-time operation), and low area and power consumption. This paper focuses on discussing the two basic approaches that can be employed to design programmable universal controller integrated circuits. Analog, mixed-signal and digital realizations are summarized and compared.

Keywords: Universal approximators, Fuzzy controllers, VLSI design, Analog and Digital.

1 Introduction

A relevant feature of fuzzy systems is that they are universal approximators and, hence, potentially suitable for any application. The problem is how to design them [1]. On one side, the numbers of inputs, outputs, and corresponding linguistic labels covering them have to be selected. On the other side, the types of membership functions, antecedents' connective and fuzzy implication as well as the methods of rule aggregation and defuzzification have to be chosen.

VLSI implementations of fuzzy systems offer the advantages of high inference speed with low area and power consumption, but they lack of flexibility, that is, the previously commented design variables are fixed in a fuzzy chip (for instance the maximum number of inputs and outputs). Programmability increases flexibility at the cost of more complex hardware. At a consequence, it is very important to select efficiently which design variables are going to be fixed and which others are to be programmable so as to achieve a good trade-off between approximation capability and hardware complexity.

Fuzzy implication and the methods of rule aggregation and defuzzification are fixed when selecting the Singleton (or zero-order Takagi-Sugeno's) Method of inference, which is the most suitable for hardware implementation. On the other

side, the number of labels per input and output depends on the architecture. Architectures based on a grid partition of the input spaces are advantageous for several reasons. Considering knowledge representation, a grid partition has semantic meaning while considering approximation theory, the problem is simplified to local piecewise interpolation. Besides, from a hardware point of view, circuitry is considerably reduced since a rule-active driven architecture can be employed. Having selected singleton fuzzy systems with a grid architecture, we will resort to approximation theory to choose the programmable parameters efficiently.

This paper is organized as follows. Section 2 summarizes the relation between Takagi-Sugeno fuzzy controllers and approximator systems. Sections 3 and 4 focus on discussing two approaches to design programmable integrated circuits for implementing universal fuzzy controllers. The objective of both approaches is to obtain general-purpose fuzzy chips that can be adjusted to a particular control application by a digitally programming interface. Their features are summarized, compared and illustrated by several examples in Section 5. Finally, conclusions are given in Section 6.

2 Fuzzy controllers as universal controllers

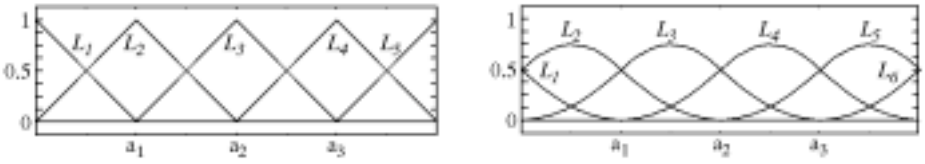
A fuzzy controller provides in general a non-linear control surface. Considering a Takagi-Sugeno fuzzy controller with a grid architecture, each input universe of discourse, I_i, is partitioned into L_i linguistic labels. If α is the maximum overlapping degree between the membership functions of the linguistic labels, $L_i + 1 - \alpha$ intervals can be distinguished per input, as can be seen in Figure 1. They are separated by the points $\{a_{i1}, a_{i2}, ..., a_{iNi}\}$, where N_i is $L_i - \alpha$. Given an input vector $\overline{x}_o = (x_{1o}, ..., x_{uo})$, the fuzzy controller identifies the u intervals, that is, the particular grid cell, GC_p, to which the input belongs. As a second step, the system provides the corresponding output $y(\overline{x}_o)$ by evaluating the α^u active rules:

$$y(\overline{x}_o) = \left. \frac{\sum_{k=1}^{\alpha^u} h_{pk}(\overline{x}) \cdot c_{pk}(\overline{x})}{\sum_{k=1}^{\alpha^u} h_{pk}(\overline{x})} \right|_{\overline{x}=\overline{x}_o} = y_p(\overline{x})|_{\overline{x}=\overline{x}_o} \quad (1)$$

where h_{pk} is the activation degree of one of the α^u active rules and c_{pk} is its corresponding consequent function.

The parameters that define the piece $y_p(\overline{x})$ are a few of the global parameters that define the fuzzy controller, $y(\overline{x})$. In this sense, a fuzzy controller is viewed as a local piecewise interpolator that provides the piece y_p for each GC_p grid cell. The interpolation provided can be piecewise constant, piecewise linear, piecewise quadratic, and piecewise nonlinear, in general, depending on whether y_p is constant, linear, quadratic, or nonlinear in $\overline{x}$. Hence, fuzzy controllers can approximate any control surface and that is why they can be called universal controllers.

When considering the implementation of complex fuzzy systems with programmable integrated circuits, singleton values are preferred for the consequents in order to reduce circuitry. Hence, this paper focuses mainly on the approximation obtained by singleton fuzzy controllers.



(a) (b)

Figure 1: Different coverings of the input space with overlapping degrees of (a) 2 and (b) 3.

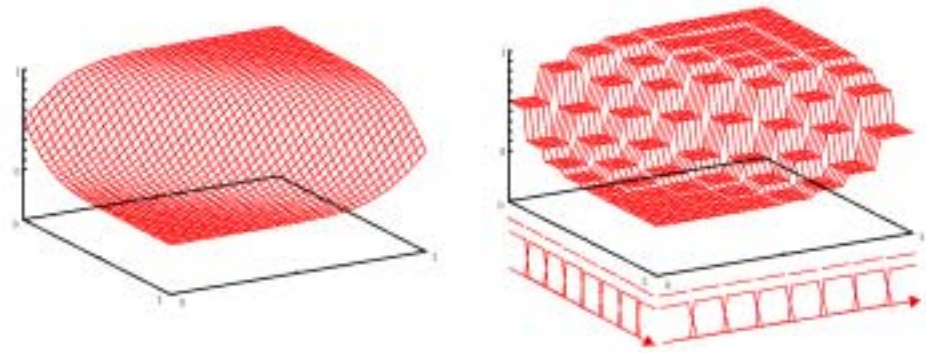
2.1 Piecewise constant controllers

If the membership functions do not overlap, the control value is directly the consequent function and there is anything fuzzy. For instance, if the consequents are constant or singleton values, the result is a piecewise constant controller.

Figure 2 illustrates an example of this type of approximation. The control surface to approximate (Figure 2a) is $f(x, y) = 1/(1+e^{10(x-y)})$. The approximation shown in Figure 2b has been obtained by a controller that divides the input space into 8x8 = 64 grid cells, that is, 8 non-overlapping membership functions per input, as can be seen at the bottom of the figure. The maximum error of this approximation is 0.29.

2.2 Piecewise multilinear controllers

If the membership functions of the rule antecedents have an overlapping degree of 2, the input space is divided into $(L-1)^u$ cells, where L is the number of membership functions covering the u input variables. A fuzzy controller provides a multilinear piece for these grid cells if the following constraints are met: (a) the membership functions are linear and normalized (that is, B-splines of degree 1 [2]



(a) (b)

Figure 2: Example of a piecewise-constant controller: (a) target; (b) approximation.

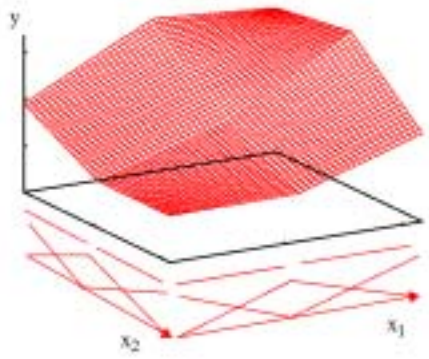


Figure 3: Example of a piecewise-linear controller.

like those in Figure 1a), (b) the antecedents' connective is the normalized product or its approximation [3], and (c) the consequents are constant. For instance, given two inputs, x_1 and x_2, and using the normalized product as antecedent connective the output piece, according to Equation (1), is given by:

$$\begin{aligned} y_p(x_1,x_2) &= c_{RR}\mu_{R1}\mu_{R2} + c_{RL}(\mu_{R1} - \mu_{R1}\mu_{R2}) + \\ &\quad + c_{LL}(1 - \mu_{R1} - \mu_{R2} + \mu_{R1}\mu_{R2}) + c_{LL}(\mu_{R2} - \mu_{R1}\mu_{R2}) \end{aligned} \quad (2)$$

where μ_{Ri} and $1 - \mu_{Ri}$ are the two non-zero membership degrees of the input x_i.

This equation can be also expressed as:

$$y_p(x_1,x_2) = ax_1x_2 + bx_2 + cx_1 + d \quad (3)$$

where a, b, c, d are constants related to the parameters of the membership functions and to the singleton values.

Compared with piecewise constant controllers, the piecewise multilinear approximators achieve a given approximation error with less input space resolution. This is shown in Figure 3, considering the same target surface as in Figure 2. The approximation illustrated has been obtained by dividing the input space into 2x2 = 4 grid cells, that is, 3 B-splines membership functions of degree 1 per input, as can be seen at the bottom of Figure 3. The maximum approximation error is 0.22.

2.3 Piecewise multiquadratic controllers

A piecewise multiquadratic controller is obtained if either the membership and consequent functions in Equation (1) are linear or the consequents are constant but the membership functions are quadratic. In the last case, the case of a singleton fuzzy controller, the overlapping degree should be 3 (B-spline membership functions of degree 2, as shown in Figure 1b) if the control surface is wanted to be continuously differentiable [4]. We will focus on the last case in the following.

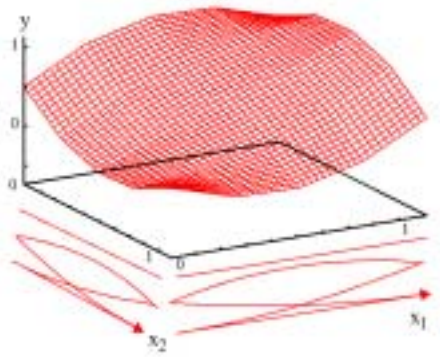


Figure 4: Example of a piecewise-biquadratic controller.

For a given input, x_i, which belongs to the interval $[a_p, a_{p+1}]$, three membership degrees can be non zero, since the overlapping degree is 3. The values of these membership degrees are as follows:

$$\begin{aligned} \mu_1 &= \frac{(x_i - a_p)^2}{(a_{p+1} - a_p) \cdot (a_{p+2} - a_p)} \\ \mu_2 &= \frac{(x_i - a_{p+1})^2}{(a_{p+1} - a_p) \cdot (a_{p+1} - a_{p-1})} \\ \mu_3 &= 1 - \mu_1 - \mu_2 \end{aligned} \tag{4}$$

Considering, for simplicity, the case of two dimensions, the output given by a fuzzy controller that employs the normalized product as antecedent connective is:

$$\begin{aligned} y_p(x_1, x_2) = {} & c_{33} + \mu_1^{x_1}(c_{13} - c_{33}) + \mu_2^{x_1}(c_{23} - c_{33}) + \mu_1^{x_2}(c_{31} - c_{33}) + \\ & + \mu_2^{x_2}(c_{32} - c_{33}) + \mu_1^{x_1}\mu_1^{x_2}(c_{11} - c_{13} - c_{31} + c_{33}) + \\ & + \mu_1^{x_1}\mu_2^{x_2}(c_{12} - c_{13} - c_{32} + c_{33}) + \\ & + \mu_2^{x_1}\mu_1^{x_2}(c_{21} - c_{23} - c_{31} + c_{33}) + \\ & + \mu_2^{x_1}\mu_2^{x_2}(c_{22} - c_{23} - c_{32} + c_{33}) \end{aligned} \tag{5}$$

where c_{ij} are the nine, α^2, singleton values activated by the input and $\mu_j^{x_i}$ takes the expressions in Equation (4).

The equation above can be expressed in function of x_1 and x_2 as follows:

$$y_p(x_1, x_2) = ax_1^2x_2^2 + bx_1x_2^2 + cx_2^2 + dx_1^2x_2 + ex_1x_2 + fx_2 + gx_1^2 + hx_1 + i \tag{6}$$

where $a, ..., i$, are constants.

Compared with piecewise multilinear controllers, piecewise multiquadratic approximators achieve a given approximation error with less input space resolution.

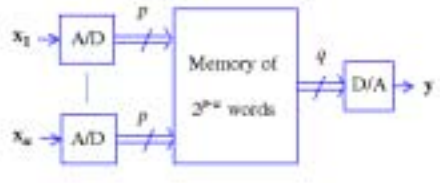


Figure 5: Look-up table realization for piecewise constant controller.

Considering the same target surface as in Figures 2 and 3, now the approximation is illustrated in Figure 4. This has been obtained by dividing the input space into 1x1=1 grid cell, that is, 3 B-splines membership functions of degree 2 per input. The maximum error is 0.15.

3 Memory-based approach

There are basically two VLSI strategies to design programmable ICs for implementing universal fuzzy controllers. One of them, which will be called memory-based approach, is to design a fuzzy chip that implements directly the piece y_p for each grid cell. This approach has been employed in digital realizations with DSPs [5] and analog ASICs [6]. Programmable parameters of this approach can be the points a_{ij} that allow identifying the active grid cell, and the constants that define the piece y_p in that grid cell. Let us analyze the implementation of different fuzzy controllers with this approach.

3.1 Piecewise constant controllers

The most direct strategy to implement a piecewise controller is to employ a look-up table that contains all the control values corresponding to every possible combination of the input signals. If there are u inputs, $\overline{x} = (x_1, ..., x_u)$, coded by p bits, a look-up table divides the input space into $2^{p \cdot u}$ grid cells and associates to each grid cell a constant control value.

The microelectronic circuits suitable to carry out this implementation are standard digital memories and programmable logic devices (PLDs). Figure 5 shows the block diagram of this type of ICs, assuming that inputs and output are analog signals. One advantage of this realization is that the response time is very short because the only operation (apart from the possible data conversions) is to retrieve a data from a memory and this can be performed in a few nanoseconds. As a drawback, look-up table implementations are practical only when the number of inputs and their quantization degrees are low, otherwise the total number of bits to store is very large.

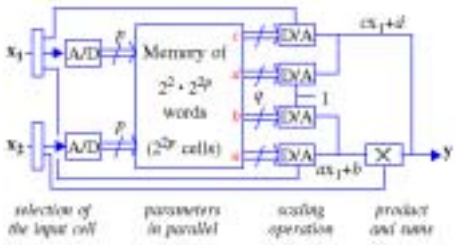


Figure 6: Memory-based parallel current-mode realization of a piecewise multilinear controller.

3.2 Piecewise multilinear controllers

A given control surface can be approximated by dividing the input space into several grid cells and associating to each cell a multilinear function, $y = y_p(\overline{x})$. As was commented above, piecewise multilinear controllers provide the following kind of pieces:

$$y_p(x_1, x_2, x_3) = ax_1x_2x_3 + bx_2x_3 + cx_1x_3 + dx_3 + ex_1x_2 + fx_2 + gx_1 + h \qquad (7)$$

in the case of three inputs, where $a, b, ..., h$ are constants that can be suitably programmed.

The memory-based implementation of programmable piecewise multilinear controllers requires to retrieve the programmable parameters from a memory and to implement multiplication and addition operations. Multiplication is more costly to implement by hardware than addition. Therefore, calculation of the output can be performed as follows to save multipliers:

$$y_p(x_1, x_2, x_3) = x_3[x_2(ax_1 + b) + cx_1 + d] + x_2(ex_1 + f) + gx_1 + h \qquad (8)$$

In this way, the operations to implement in the case of u inputs are: (a) $2^u - 1$ additions; (b) a set of 2^{u-1} constants that have to be retrieved from memory; (c) another set of 2^{u-1} constants that have to be retrieved from memory and multiplied by variables; and (d) $2^{u-1} - 1$ multiplications between variables.

A parallel realization of these operations offers the shortest processing time but the largest circuitry. For instance, the memory that contains the programmable parameters $(a, b, ..., h$, in Equations (7) and (8)) must have 2^u output buses. In the case of analog current-mode circuitry, the additions can be performed by simply connecting wires; and the following blocks are required: 2^u D/A converters (like programmable current mirrors), to convert and to scale the two sets of 2^{u-1} constants; and $2^{u-1} - 1$ multipliers. The block diagram of this type of realization for the case of two inputs is shown in Figure 6. In the case of digital circuitry, $2^u - 1$ adders and $2^u - 1$ multipliers are required by a parallel realization.

The circuitry can be reduced by sequentializing the operations. For instance, if we consider digital circuitry, operations can be performed in $2^u - 1$ steps by using

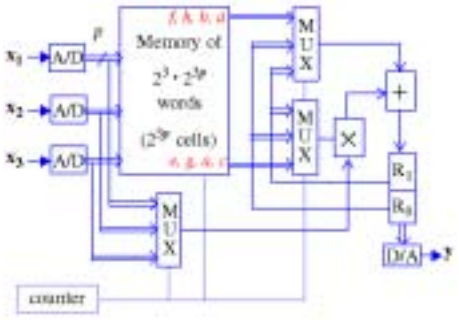


Figure 7: Memory-based sequential digital realization of a piecewise multilinear controller.

1 adder, 1 multiplier, and a memory with 2 output buses. The block diagram of this type of realization for the case of three inputs is shown in Figure 7, where R_1 and R_0 are two registers. In this case, the operation sequence is: $cx_1 + d$, $ax_1 + b$, $R_1x_2 + R_0$, $gx_1 + h$, $R_0x_3 + R_1$, $ex_1 + f$, and $R_1x_2 + R_0$.

3.3 Piecewise quadratic controllers

As was commented in Section 2, piecewise multiquadratic controllers can be implemented by dividing the input space into several grid cells and associating to each cell a multiquadratic function, $y = y_p(\overline{x})$. In the case of two inputs these pieces are as follows:

$$y_p(x_1, x_2) = ax_1^2x_2^2 + bx_1x_2^2 + cx_2^2 + dx_1^2x_2 + ex_1x_2 + fx_2 + gx_1^2 + hx_1 + i \quad (9)$$

where $a, b, ..., i$ are constants that can be suitably programmed.

The memory-based implementation of programmable piecewise multiquadratic controllers requires to retrieve the programmable parameters from a memory and to implement not only multiplication and addition but also squaring operations. To save multipliers, calculation of the output can be performed as follows:

$$y_p(x_1, x_2) = x_2^2(ax_1^2 + bx_1 + c) + x_2(dx_1^2 + ex_1 + f) + gx_1^2 + hx_1 + i \quad (10)$$

in the case of two inputs.

The operations to implement in the case of u inputs are: (a) $3^u - 1$ additions; (b) a set of 3^{u-1} constants that have to be retrieved from memory; (c) two sets of 3^{u-1} constants that have to be retrieved from memory and multiplied by variables; (d) $2^u - 2$ multiplications between variables; and (e) u squaring operations.

For a parallel realization, the memory that contains the programmable parameters ($a, ..., i$, in Equations (9) and (10)) must have 3^u output buses. The following blocks are required when using analog current-mode circuitry: 3^u D/A converters (like programmable current mirrors), to convert and to scale the three sets of 3^{u-1}

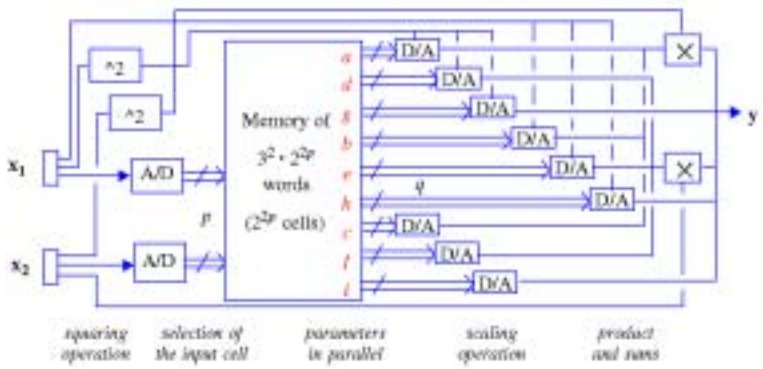


Figure 8: Memory-based parallel current-mode realization of a piecewise multiquadratic controller.

constants; and $2^u - 2$ multipliers. The block diagram of this type of realization for the case of two inputs is shown in Figure 8. If digital circuitry is employed, $3^u - 1$ adders and $2 \cdot 3^{u-1} + 2^u - 2$ multipliers are required by a parallel realization. Circuitry can be reduced by sequentializing the operations.

4 MFC-based approach

Instead of implementing directly the expression of each output piece, the MFC-based approach implements the different stages of a fuzzy inference mechanism. These stages are: calculation of the membership degrees,$\mu_j(x_i)$, (by membership function circuits, MFCs), calculation of the activation degrees of the rules, $h_{pk}(\overline{x})$, (by connective circuits), and computation of a weighted average, according to Equation (1). This approach will be called MFC-based approach because its main difference with the previous one is that membership degrees are calculated explicitly. This approach has been followed by many analog, digital and mixed-signal fuzzy ASICs [7].

Circuitry can be simplified if certain constraints are imposed. In this sense, the membership functions that cover each input universe of discourse usually overlap each other with a degree,α, of 2 or 3. This happens, for instance, when B-splines of degree 1 and 2 are selected as membership functions (Figure 1a and b, respectively). This means that 2 or 3 MFCs per input are required to generate the membership functions. Another constraint is that the membership functions are preferred to be normalized, that is, the overlapped membership functions verify that their sum is constant (as shown in Figure 1). In this case, $\alpha - 1$ MFCs per input are required because the other membership degree can be calculated with an addition.

If normalized membership functions are employed and if the normalized product or a piecewise-linear approximation to the product [3] is chosen as the connective operator, the denominator of the weighted average in Equation (1) is constant, so that no divider is required.

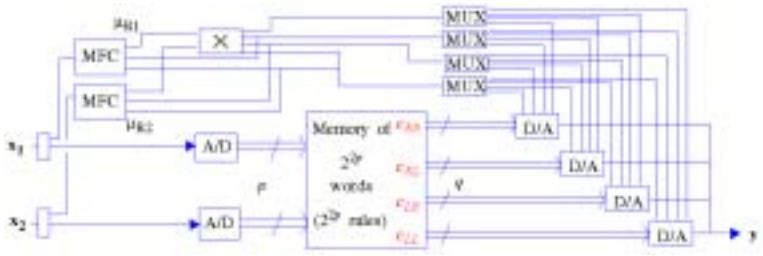


Figure 9: MFC-based parallel current-mode realization of a piecewise multilinear controller.

In addition, hardware is simplified, specially in this MFC-based approach, if all the intervals defined by the membership functions in the input signals have the same width, $h = a_{p+1} - a_p$, because the denominators in the expressions of the membership degrees are constant so that divider circuits are neither required at the fuzzification stage.

4.1 Piecewise constant controllers

Since in this case there is anything fuzzy, the microelectronic design can not follow the MFC-based approach. The design would be equal to that discussed in the memory-based approach.

4.2 Piecewise multilinear controllers

As was commented in Section 2, given two inputs, x_1 and x_2, and using the normalized product as antecedent connective the output of a piecewise multilinear controller is given by:

$$\begin{aligned} y_p(x_1, x_2) &= c_{RR}\mu_{R1}\mu_{R2} + c_{RL}(\mu_{R1} - \mu_{R1}\mu_{R2}) + \\ &\quad + c_{LL}(1 - \mu_{R1} - \mu_{R2} + \mu_{R1}\mu_{R2}) + c_{LL}(\mu_{R2} - \mu_{R1}\mu_{R2}) \end{aligned} \quad (11)$$

where μ_{Ri} and $1 - \mu_{Ri}$ are the two non-zero membership degrees of the input x_i.

Similar to the memory-based approach, this MFC-based approach requires also $2^{u-1} - 1$ multiplications between two variables and to retrieve 2^u constants from a memory. An important difference is that the number of words to store is now $2^{M \cdot u}$, while it is $2^{(N+1) \cdot u}$ in the memory-based approach, considering input space partitions of $(2^M - 1)^u$ and $2^{N \cdot u}$ grid cells, respectively. This means that, for equal input space partitions, the memory size in the MFC-based approach is almost half of the memory size in the other approach. The reason is that each word of the memory in the MFC-based strategy is used by 2^u different grid cells while in the other approach it is used by only 1 cell. The cost to pay for this memory reduction is to include several multiplexors and, in this case, 1 MFC for each input variable.

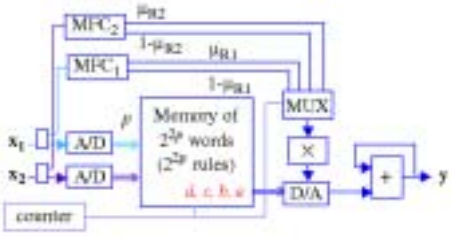


Figure 10: MFC-based sequential digital realization of a piecewise multilinear controller.

For instance, to implement the piecewise bilinear controller of Figure 3 with the MFC-based approach, a memory with 9 words is required (9 singleton values for the 9 rules), while using the memory-based approach, the memory must store 16 parameters (4 parameters for the 4 grid cells).

A parallel current-mode realization following this MFC-based strategy is shown in Figure 9 for the case of two inputs. In a parallel realization with u inputs, the memory of $2^{M \cdot u}$ words is divided into 2^u parts, each one containing $2^{(M-1)u}$ words that are never required simultaneously. Multiplexors have to be added to select which of the 2^u parts provides each of the 2^u words required [7]. The circuitry can be reduced by sequentializing the operations. For instance, operations can be performed in 2^u steps by using 1 accumulator, 1 multiplier, and a memory with 1 output bus [7]. Figure 10 illustrates the block diagram of this sequential realization for the case of two inputs.

4.3 Piecewise multiquadratic controllers

As was commented in Section 2, a fuzzy controller with B-spline membership functions of degree 2 connected by the product and with constant consequents acts as a piecewise multiquadratic controller. It provides the following output piece, in the case of 2 inputs (Equation (5)):

$$\begin{aligned} y_p(x_1, x_2) \;=\; & c_{33} + \mu_1^{x_1}(c_{13} - c_{33}) + \mu_2^{x_1}(c_{23} - c_{33}) + \mu_1^{x_2}(c_{31} - c_{33}) + \\ & + \mu_2^{x_2}(c_{32} - c_{33}) + \mu_1^{x_1}\mu_1^{x_2}(c_{11} - c_{13} - c_{31} + c_{33}) + \\ & + \mu_1^{x_1}\mu_2^{x_2}(c_{12} - c_{13} - c_{32} + c_{33}) + \\ & + \mu_2^{x_1}\mu_1^{x_2}(c_{21} - c_{23} - c_{31} + c_{33}) + \\ & + \mu_2^{x_1}\mu_2^{x_2}(c_{22} - c_{23} - c_{32} + c_{33}) \end{aligned} \tag{12}$$

where c_{ij} are the nine singleton values activated by the input.

A parallel current-mode realization that implements the above equation requires the following hardware: (a) a global memory that stores $2^{M \cdot u}$ words, where 2^M is the number of membership functions per input; (b) u A/D converters that address the memory; (c) $2^u - 2$ analog multipliers (by grouping the signals conveniently);

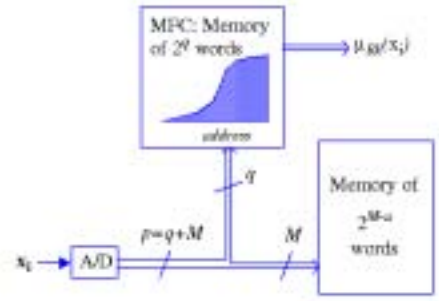


Figure 11: Memory-based MFC that generates a generic membership function for an input variable.

(d) 3^u D/A converters to provide the scaling with the singleton values; and (e) $2u$ MFCs (squaring circuits) that provide the membership degrees (Equation (4)):

$$\begin{aligned} \mu_1 &= \frac{(x_i - a_p)^2}{(a_{p+1} - a_p) \cdot (a_{p+2} - a_p)} = \frac{(x_i - a_p)^2}{2h^2} \\ \mu_2 &= \frac{(x_i - a_{p+1})^2}{(a_{p+1} - a_p) \cdot (a_{p+1} - a_{p-1})} = \frac{(x_i - a_{p+1})^2}{2h^2} \end{aligned} \tag{13}$$

where h is the constant width of the intervals that the membership functions define in the input universes of discourse.

In a parallel realization, the global memory should be divided into 3^u parts to retrieve the required parameters in just one clock cycle.

4.4 Approaches to implement the MFCs

There are three basic approaches to implement the MFCs. One approach is to generate all the membership functions of a variable by means of a memory. This is the most versatile approach because it allows different shapes for all the membership functions (bell-shaped, triangular, trapezoidal, etc.).

A second approach is to generate the same shape for all the membership functions of an input variable by means of a memory. In this case, the first operation is to identify the input interval $[a_p, a_{p+1}]$ to which the input belongs. The code to address the memory is obtained by an A/D converter whose input signal is $x_i - a_p$ and its reference signal is $h = a_{p+1} - a_p$. This approach is illustrated in Figure 11.

The third approach is to generate the same shape for all the membership functions of an input variable by means of an arithmetic block. The first operation to perform, like in the previous approach, is to identify the input interval $[a_p, a_{p+1}]$ to which the input belongs. These breakpoints are input signals for the arithmetic MFC. Some examples are digital or current-mode analog MFCs that generate triangular membership functions and transconductance-mode analog MFCs, based on differential pairs, that generate bell-shaped membership functions [7]. In the

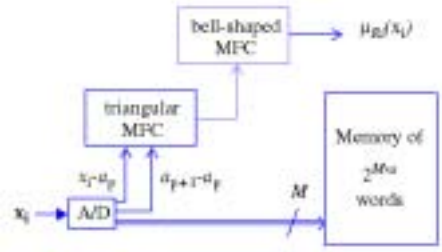


Figure 12: Arithmetic MFC that generates a generic membership function for an input variable.

case of bell-shaped MFCs, a triangular MFC can be used to displace and scale the input, as illustrated in Figure 12.

5 Examples

Among the two approaches described herein, the memory-based approach is the most versatile and the easiest strategy to implement but it is the most costly if the number of words to store is great. On the other side, the MFC-based approach is less versatile and more difficult but it is the least costly in terms of silicon area consumption. Given a range of applications, the designer has to select the adequate strategy and hence to explore a wide design space. A CAD tool is very helpful to assist the designer in this choice. We have developed a tool called *fmcsim* that simulates at a behavioral level the performance of the different alternatives [8]. Given a desired input-output mapping to reproduce, this tools allows: (a) to compare the output provided by the different strategies depending on the input partitions, the resolution of the building blocks employed (memories and computing blocks), the types of MFCs and connective operators (in the case of MFC-based implementation of fuzzy systems), and (b) to find the adequate input partitions and the adequate values for the parameters that define the output pieces, taking into account the resolution of the building blocks.

As examples, we have studied the approximation of the following surfaces:

$$F_1 : z = \frac{1}{1 + e^{10(x-y)}} \tag{14}$$

$$F_2 : z = 0.5(1 + \sin(2\pi x)\cos(2\pi y)) \tag{15}$$

F1 is the surface that was illustrated in Figure 2a. The surface F2 is shown in Figure 13.

Tables 1 and 2 shows some of the information obtained with this tool when comparing the different strategies. Both sets of results consider a resolution of 8 bits for the memories and computing blocks, except for the memory-based MFCs, which consider resolutions of 2 (Table 1) and 3 bits (Table 2). Some of the approximations

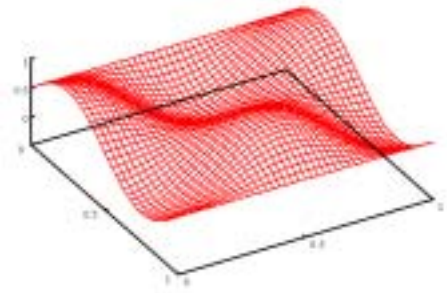


Figure 13: Target surface F_2.

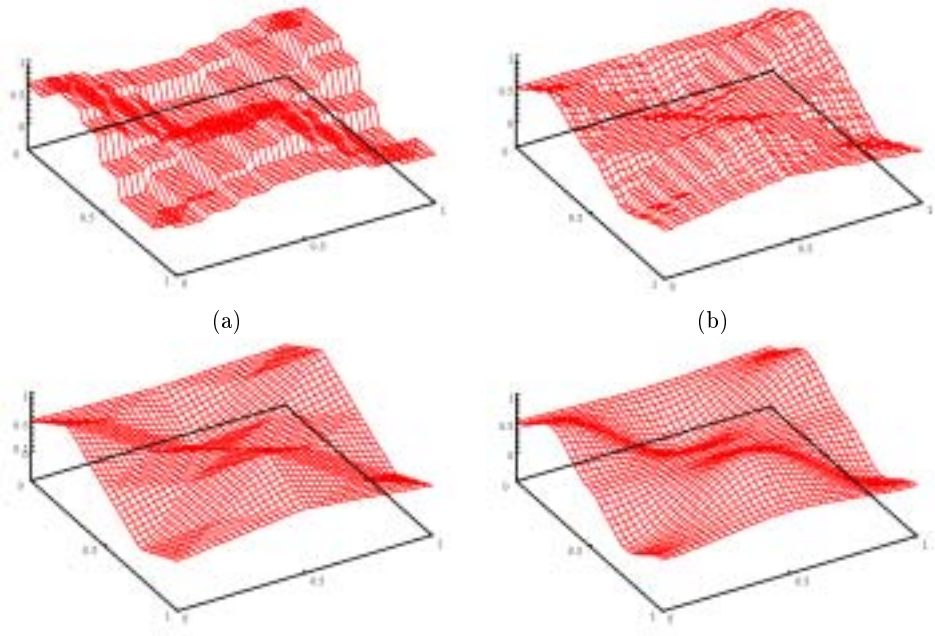
obtained for the F2 function (which appear with bold characters in Table 2) are illustrated in Figure 14.

Table 1: Results when approximating F_1.

Approach	Memory-based		MFC-based					
type of approx.	piecewise constant	piecewise linear	piecewise linear (digital MFCs)		piecewise linear (analog and triangular MFC)		piecewise non-linear (analog and bell-shaped MFC)	
antecedent connective			product	approx.	product	approx.	product	approx.
memory words	8x8x1=64	4x2x4=32	2x4+16=24		4x4=16		4x4=16	
maximum error	0.29	0.18	0.26	0.26	0.13	0.16	0.19	0.22
RMSE(%)	7.77	7.44	8.14	8.19	6.28	6.20	6.46	6.33

Table 2: Results when approximating F_2.

Approach	Memory-based		MFC-based					
type of approx.	piecewise constant	piecewise linear	piecewise linear (digital MFCs)		piecewise linear (analog and triangular MFC)		piecewise non-linear (analog and bell-shaped MFC)	
antecedent connective			product	approx.	product	approx.	product	approx.
memory words	8x8x1=64	4x2x4=32	2x8+15=24		5x3=15		5x3=15	
maximum error	0.24	0.16	0.27	0.26	0.16	0.15	0.13	0.14
RMSE(%)	**8.89** (Fig. 14a)	8.37	10.91	**10.48** (Fig. 14b)	8.37	**8.07** (Fig. 14c)	**5.36** (Fig. 14d)	5.81



(a) (b)

(c) (d)

Figure 14: Simulated performance of different VLSI implementations of fuzzy controllers that approximate the target surface in Figure 13.

6 Conclusions

Two different strategies (memory- and MFC-based) have been described and compared for the VLSI design of universal fuzzy controllers. They differ in the type of hardware resources employed (more use of memory against computing blocks or vice versa) and in the system performance achieved (more flexibility against simplicity or vice versa). Piecewise constant, multilinear, multiquadratic, and non-linear controllers have been discussed. Analog, digital, and mixed-signal realizations have been illustrated. The use of digital computing blocks introduces quantization errors that usually dominate the resulting approximation error.

The first operation to carry out in any strategy is to identify which cell of the input space is activated by the inputs to correctly address the memory. If all the cells have the same size, h (the partition of the input spaces is homogeneous), A/D converters are employed. In general, approximation degree is increased by allowing this partition to be non-homogeneous. In this case, the breakpoints of each input space have to be stored and the A/D converters are not standard converters but blocks that compare the inputs with the breakpoints instead of scaled versions of a reference signal. Considering hardware implementations, the designer has to evaluate if the approximation improvement justify the increase of hardware complexity. A CAD tool has been developed to assist the designer in the different choices.

References

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

[2] M. J. D. Powell, "*Approximation theory and methods*", Cambridge University Press, 1981.

[3] R. Rovatti, "Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in Sobolev norms", *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 2, pp. 235-249, 1998.

[4] X.-J. Zeng, M. G. Singh, "Approximation accuracy analysis of fuzzy systems as function approximators", *IEEE Trans. on Fuzzy Systems*, Vol. 4, No. 1, pp. 44-63, Feb. 1996.

[5] R. Rovatti, M. Vittuari, "Linear and fuzzy piecewise-linear signal processing with an extended DSP architecture", in *Proc. IEEE Int. Conference on Fuzzy Systems*, pp. 1082-1087, Anchorage, 1998.

[6] J. Matas, L. García de Vicuña, M. Castilla, "A synthesis of fuzzy control surfaces in CMOS technologies", in *Proc. IEEE Int. Conference on Fuzzy Systems*, Vol. 2, pp. 641-647, Barcelona, 1997.

[7] I. Baturone, A. Barriga, S. Sánchez-Solano, C. J. Jiménez-Fernández, and D. R. López, *Microelectronic design of fuzzy logic-based systems*, CRC Press, 2000.

[8] G. Hernández Pastelero, "Fmcsim, un simulador para chips mixtos difusos", Proyecto Fin de Carrera, Ing. Informática, Univ. de Sevilla, Dic. 1999.