A Multiplexed Mixed-Signal Fuzzy Architecture

Fernando Vidal-Verdú, Rafael Navas-González, Universidad de Málaga, SPAIN
Angel Rodríguez-Vázquez, Centro Nacional de Microelectrónica-Universidad de Sevilla, SPAIN

Abstract:

Analog circuits provide better area/power efficiency than their digital counterparts for low-medium precision requirements [1]. This limit in precision, as well as the lack of design tools when compared to the digital approach, imposes a limit of complexity, hence fuzzy analog controllers are usually oriented to fast low-power systems with low-medium complexity. This paper presents a strategy to preserve most of the advantages of an analog implementation, while allowing a notorious increment of the system complexity. Such strategy consists in implementing a reduced number of rules, those that really determine the output in a lattice controller, which we call analog core, then this core is dynamically programmed to perform the computation related to a specific rule set. The data to program the analog core are stored in a memory, and constitutes the whole knowledge base in a kind of virtual rule set. HSPICE simulations from an exemplary controller are shown to illustrate the viability of the proposal.

1. Introduction

The widespread interest in fuzzy logic technology motivates the implementation of special purpose ASICs in the electronics design field, which speed up the computation of the fuzzy algorithm when required [2][3]. Such circuits can be digital [4] or analog [5][6]. Pipeline techniques in the former allow to compute a very high number of fuzzy rules per second, because the bandwidth is limited by the delay of the standard cell in the used technology. However, the input-output delay, which is indeed the most important parameter in real time control, is usually higher in the digital than in the analog implementations. Analog implementations usually reports low power and area consumption, and offer a natural interface to sensors and actuators, because they do not need converters in the signal path. In addition, the low resolution, their usual drawback, does not seem a major problem in fuzzy control applications, most of them working below four bits [6]. However, since the fuzzy algorithm involves global computation steps [7][8], like the center of gravity, errors due to systematic or random causes are aggregated in global computation nodes, so eventually the total error becomes too large. This is specially true in control applications, where lattice partitions of the universe of discourse are common [8]. Such partitions suffer from the *curse of dimensionality*, i.e. the number of rules grows exponentially with the number of inputs, then the area and power consumption. Thus, for medium complex systems analog implementations begin to have problems to be applied.

Recently, Masetti et al. have presented a digital architecture that computes the system output only from the set of rules that have a certain influence on it, that is those rules with non zero output [9]. Such approach allows to decrease the input-output delay by exploiting the local feature of fuzzy basis functions. This paper presents a mixed signal controller that is based on the same idea, but performing the computations with analog circuits, thus mostly preserving the advantages of analog implementations. The presented controller has an analog core that implements the rules that have influence on an interpolation interval, which could be called the *real* rule set, then multiplexes such core to provide an output for any input value, among the whole knowledge which is composed by the *virtual* rule set. The latter is actually just a set of programming parameters stored in digital and analog memories, and the multiplexing is realized by means of digital techniques.

2. Architecture and Functional Description

The key of the proposed strategy lies on the concept of active rule, as the rule that contributes to determine the controller output. In a lattice partition like that depicted in Fig.1, for a bidimensional universe of discourse, any input pair (x_1,x_2) in the light shaded interval $C_{ij} = [(\varepsilon_{1i},\varepsilon_{1i+1}), (\varepsilon_{2j},\varepsilon_{2j+1})]$ maps into an output determined by the rules enclosed in the dark shaded interval, which are called

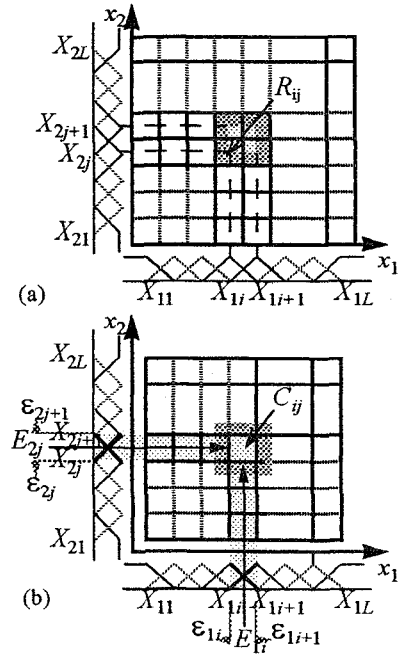

Fig. 1: Illustration of the active rule set: (a) Lattice Partition; (b) Interpolation Intervals

0-7803-4863-X/98 $10.00©1998 IEEE

active rules. Any other rule of the knowledge base have a null influence on the controller output. Fig.1(b) shows the universe of discourse split into interpolation intervals, that is, into intervals whose active rules set is different. In the interpolation procedure [8], only the membership functions associated to the active rules, and their associated singleton values, are needed. Note also that only the membership function piece that affects the output is needed. Thus, in the case illustrated in Fig.1(b), only the thick pieces of the membership functions associated to the labels X_{1i}, X_{1i+1}, X_{2j} and X_{2j+1}, and the singletons associated to the four active rules consequents, $\mathbf{y}^*_{ij}$, $\mathbf{y}^*_{i(j+1)}$, $\mathbf{y}^*_{(i+1)j}$ and $\mathbf{y}^*_{(i+1)(j+1)}$ are needed to generate the output in the interval C_{ij}.

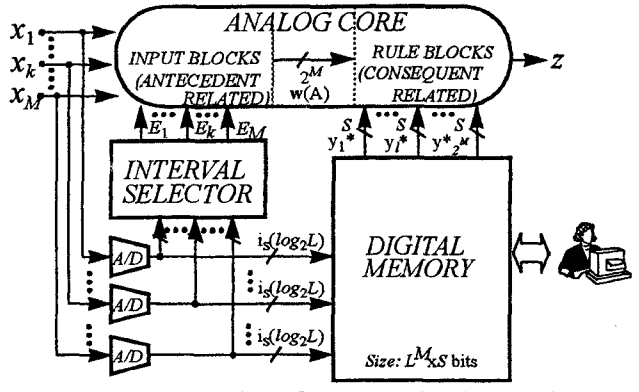

Fig. 2: General Architecture of a Controller with Multiplexed Analog Core.

The Fig.2 shows a general architecture able to perform the above procedure for a controller with M inputs and L labels per input (thus L^M rules), which involves the following high level building blocks:

- *Analog core*: this block performs the fuzzy computation. It has a set of programming inputs which are driven by the *interval selector* and the *digital memory* blocks. These inputs set up the analog core to work with the rule set that determines the system output, which means to specify the membership functions associated to the rule antecedents as well as the singleton values related to the consequents.
- *A/D Converters*: these blocks allow to identify the interpolation interval C_{ij} associated to a given input. They are M analog to digital converters, one per input, with a resolution of $i_s(log_2 L)$ bits (that is, the next superior integer of $log_2 L$). Thus, they provide a word of $Mi_s(log_2 L)$ bits, which encodes each interpolation interval. This output digital word is used later to select the membership functions in the antecedent, as well as the singleton values associated to the consequent of the active rules.
- *Interval selector*: this block selects, from the digital word provided by the converters, the voltage values $E_1, \ldots E_k, \ldots E_M$ that set up the *input block* of the *analog core* to operate with the membership functions associated to the active rules in such interval.
- *Digital Memory*: this block selects, from the digital word provided by the converters, the singleton programming values $y_1^*, \ldots, y_l^*, \ldots y_{2^M}^*$ that configure the *rule block* of the *analog core* consequents of the active rules. These are digital words of as many bits as needed to encode the required set of singleton values. The memory is externally addressable to read and write accesses for programming purposes.

Note that Fig.2 is valid and even more useful for an increasing number of inputs and labels. The number of outputs can also be higher with little effort because many blocks can be shared by the circuitry dedicated to generate each output. Specifically, each additional output involves one digital memory block, and 2^M rule blocks.

3. Functional blocks implementation

The previous section describes the functionality of the blocks in Fig.2 Here we will propose CMOS implementations for such blocks.

3.1. Analog core

The architecture of the analog core is shown in Fig.3. It is the same architecture than that of the fuzzy controller described in [10], but just 2^M rules are needed here, and only two membership functions per input. Building blocks are also quite similar than those described in [10], where the reader can find more details.

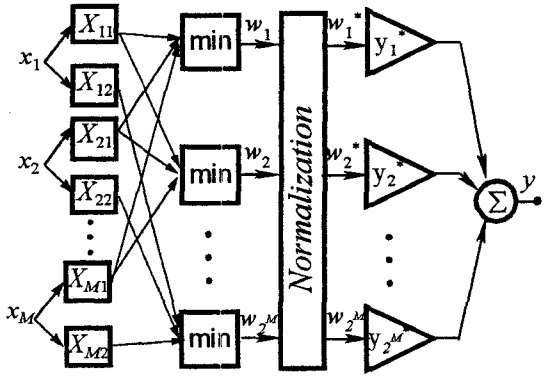

Fig. 3: Analog Core Architecture

However, some differences derived from the specific multiplexing strategy are convenient to be highlighted here. First, as said in the section 2., only the thick part of the membership functions in Fig.1(b) are needed in each interval. Hence, our membership function circuit must be able to generate two pieces with slopes of opposite signs. This is made in the simplest way by a differential pair, as Fig.4(a) depicts. Moreover, the differential pair provides two complementary curves, which can be exploited to save the explicit complement implementation if we perform the minimum by means of a maximum plus complement circuitry regarding the De Morgan's law [5]. Fig.4(b) shows one high level building block that implements most of the circuitry associated to each input in the first and second layers of Fig.3. The proposed implementation has voltages as inputs, while further processing is made in current mode. Current outputs of the differential pair are replicated to generate 2^M outputs required to implement the 2^M rules that determine the output inside a specific interval. Such currents are converted into voltages by the minimum circuit input unit cell that is shaded in Fig.4(b). The 2^M

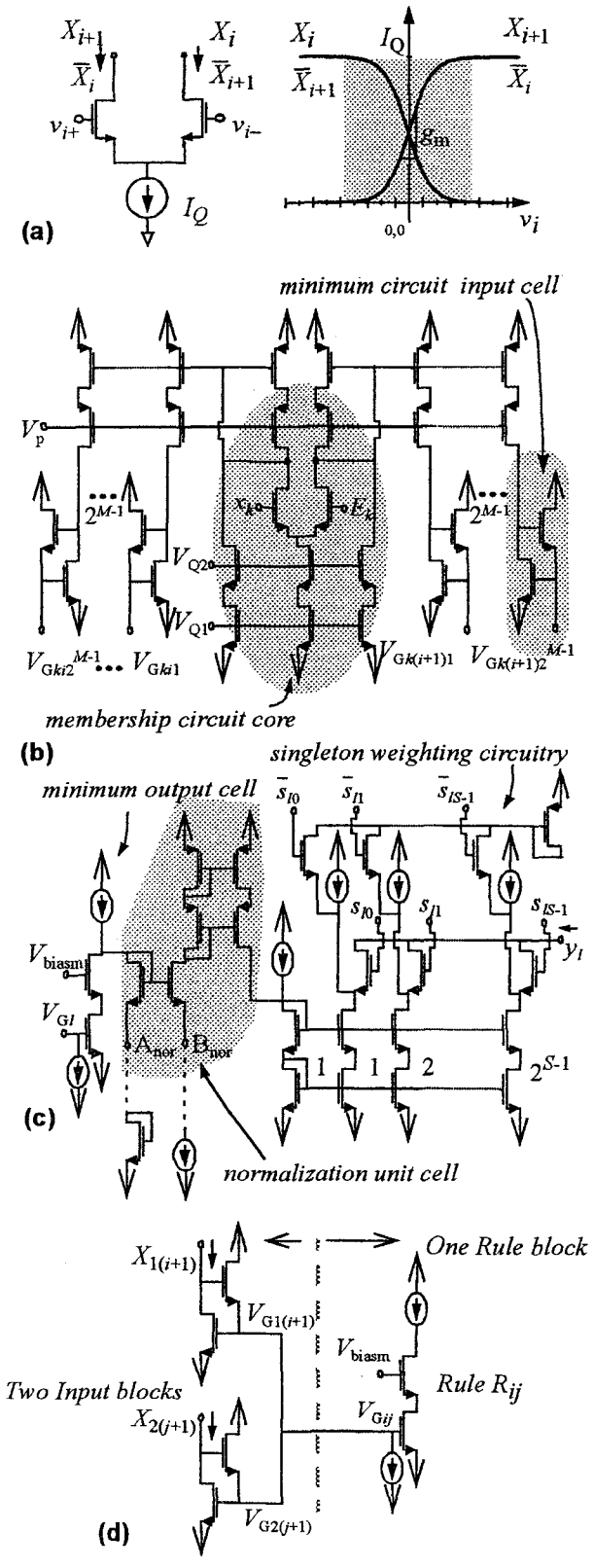

Fig. 4: Analog Core Implementation; (a) Membership Function Generation, (b) Input Block and (c) Rule Block (d) Interface

voltage outputs of this block are attached to those provided by the remaining input blocks. As a result, 2^M rule antecedents are implemented. The voltage outputs of these antecedents feed 2^M rule blocks like that depicted in Fig.4(c). Circuitry in Fig.4(c) implements the blocks in the third and fourth layers of Fig.3, as well as the minimum circuit output stage. This stage belongs to the minimum circuit that implements the minimum block in Fig.3, which actually acts as the interface between the higher level input and rule blocks. To illustrate this, Fig.4(d) shows an exemplary interface to build the rule R_{ij} in Fig.1. In the example of Fig.1, we need 2 input blocks, which provide four voltage outputs each (note that the differential pair outputs are replicated to share the circuit and save area and power consumption). To build the rule R_{ij}, the outputs $V_{G1(i+1)}$ and $V_{G2(j+1)}$ corresponding to the membership functions $X_{1(i+1)}$ and $X_{2(j+1)}$, are connected to the minimum output cell of the rule block associated to R_{ij}. Fig.4(d) is in fact a minimum circuit where the complement at input is saved because the complements are directly provided. Note that $X_{1(i+1)}$ and $X_{2(j+1)}$ are the complements of X_{1i} and X_{2j} respectively in the interpolation interval C_{ij} of Fig.1.

The high level block in Fig.4(c) is very similar to the rule block described in [10]. However, the singleton circuitry is slightly different. Since such cell is set up dynamically by the digital word $s_{I0}...s_{IS-1}$, the changes of this word generate transitories in the output current. Such transitories looked like quite large glitches in the output current. The main cause of such glitches is the current demanded by branches that do not contribute to the output current, and whose transistors enter in ohmic region, when they are selected again to report some current to the output. A proposed solution keeps all transistors in saturation by providing an alternative current path through current switches driven by the complementary control signals $\bar{s}_{I0}...\bar{s}_{IS-1}$. A higher power consumption is, again, the price to pay for a better dynamic behavior. The system global output is obtained by aggregating the rule blocks outputs,

$$y = \sum_{I=1...2^L} y_I \qquad (1)$$

which is realized just by attaching the rule block output nodes, because the rule block outputs are currents.

3.2. A/D Converters

Many possible implementations of A/D converters have been reported and could be used for this block. However, since a resolution of $i_s(log_2 L)$ bits is required and L (number of labels) rarely is higher than seven, a simple and fast flash converter like that depicted in Fig.5 can be used. Although it is a common flash converter, some details are worthy to be commented here about its implementation. First, the array of linear resistors generates twice voltage levels than those needed for the A/D conversion. The'extra' voltage levels are used as programming values for the rule antecedent, thus they are a kind of analog read only memory. Note also that this array of resistors can be shared by all the converters (one per input) as long as the comparators have high impedance inputs. Second, these comparators are designed to have hysteresis, which is needed to filter the noise associated to the system inputs and avoid an unstable output due to unstable programming

inputs to the analog core. Finally, a Gray coder converts a thermometer scale into Gray code. Such output code is used to minimize the transitions between logical values '0' and '1' in the interconnection lines, which avoids risks of spurious data due to the asynchronous operation and minimizes the noise injected in the remaining circuitry from these lines.

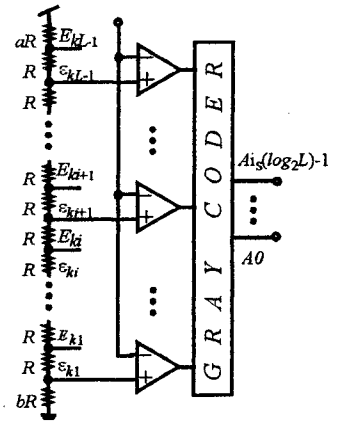

Fig. 5: A/D Flash Converter

3.3. Interval selector

The interval selector block basically is an analog bus whose analog data are selected digitally from the set of reference voltages generated by the linear resistor array in Fig.3(a), $E_1 \ldots E_M$. The Fig.6 shows one of the M cells (one per input) that constitute this block. A Gray decoder provides the control signals that drive the transmission gates form the digital word associated to an specific interpolation interval and supplied by the A/D converters.

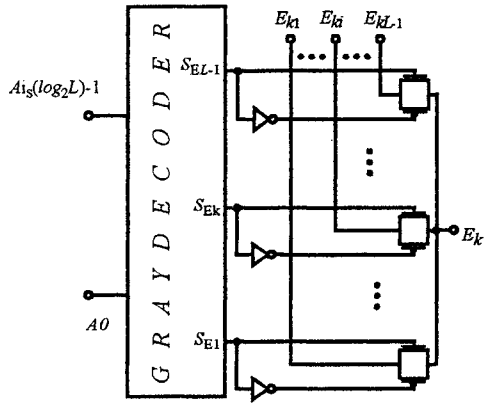

Fig. 6: Interval Selector *k* Cell

3.4. Digital memory

The digital word of $M \times i_s(\log_2 L)$ bits provided by the A/D converters and associated to each interval in the input space is used to address a digital memory. This memory must provide the singleton values that are needed to generate the output in each interval. Thus, the memory must provide a word of 2^M x S bits, where S is the number of bits per singleton value and 2^M is the number of singleton values that are necessary to program the analog core. It is

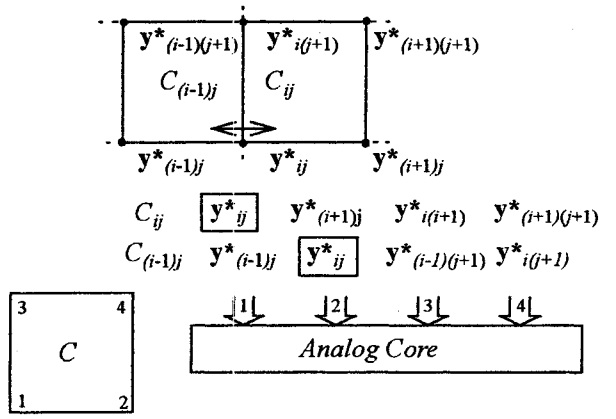

Fig. 7: Illustration of Analog Core Programming

very important to note that the order of these singleton values inside the word must fulfill the programming requirements of the analog core. To explain it, let us show the simple case of Fig.7, where the words needed to program the analog core in two adjacent intervals share two singleton values ($\mathbf{y}^*_{ij}$ and $\mathbf{y}^*_{i(j+1)}$), but these singleton values appear in different order in these words. A possible strategy to implement the memory consists in storing one word for each interval with the singleton values associated to it in the proper location. However, this implies to replicate each singleton value as many times as intervals are related to it, which is 2^M. A second strategy organizes properly the data and multiplexes the memory output bus to avoid redundancy, which is the strategy followed by the exemplary controller of this paper. The multiplexing is not made in time, the required data are put in the output bus in one step. The Fig.8 shows the proposed memory architecture that follows the latter approach. Singleton values are distributed into 2^{M-1} memory cell arrays of $L^{M-1}/2^{M-1}$ rows and L columns. Every memory cell

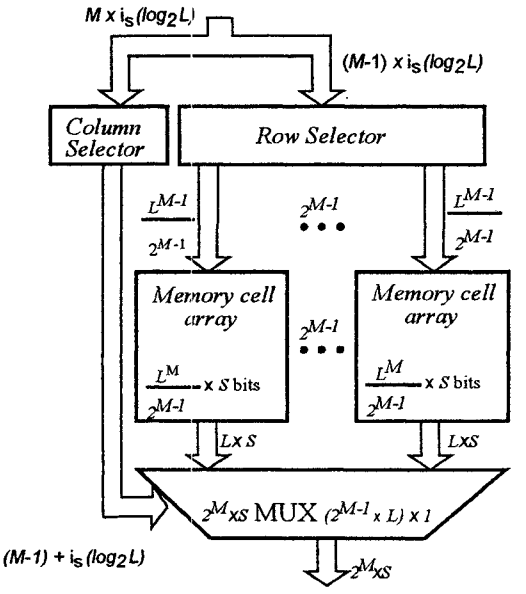

Fig. 8: Digital memory architecture

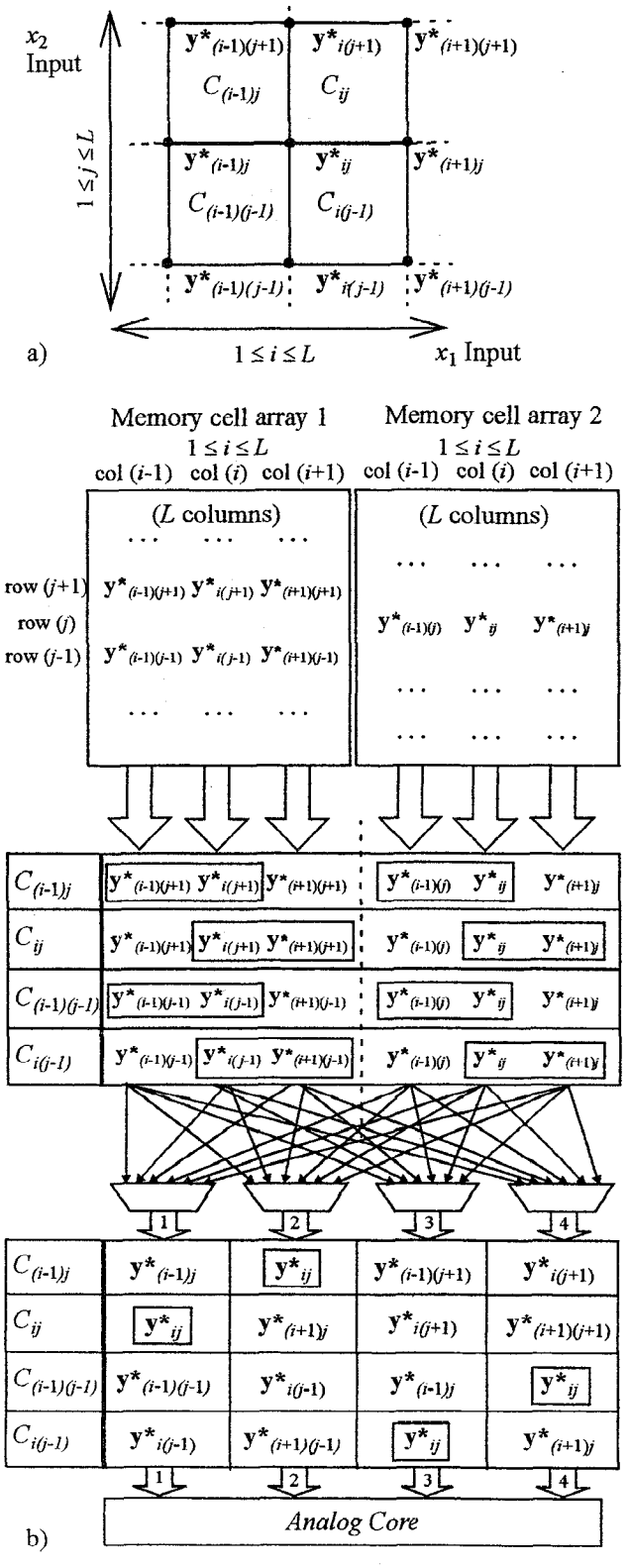

Fig. 9: Memory operation example

array stores pairs of singleton values which are needed to be addressed simultaneously. The row selector selects 2^{M-1} rows simultaneously, one for each memory cell array, to provid a set of $L \times 2^{M-1}$ singleton values. The column selector selects the correct singleton values from this set, and controls the multiplexor to place them in the right location to form the 2^M x S output word, all in one step. To illustrate the data distribution and memory operation let us to show the situation for a bidimensional case. The Fig.9(a) shows four generic adjacent intervals C_{ij} ,$C_{(i-1)j}$,$C_{(i-1)(j-1)}$,$C_{i(j-1)}$ with their associated singleton values. These values are distributed into two memory cell arrays as the top of Fig.9(b) shows. The memory cell array 1 contains the rows with odd j index, while the memory cell array 2 contains the rows with even j index. Both arrays have L columns. Below these arrays in Fig.9(b) the result of a row selection is shown for the four intervals in Fig.9(a). All the singleton values needed to program the analog core (enclosed in squares in the figure) are in the resulting word. Finally the column selector selects these singleton values and places them in the output bus. Note that each singleton value appears in the right location (see Fig.7). Although some tricky for internal accesses, the memory is configured as a conventional RAM for external accesses.

4. Results and conclusions

This paper is intended to present a strategy to implement high-medium complexity controllers while taking advantage from analog design. An exemplary controller has been designed and simulated to demonstrate the viability of the proposal. Some performance aspects of such controller are being improved currently to be introduced in a silicon prototype. The controller was designed with a CMOS 0.7μm technology and here we will show some results from HSPICE simulations to illustrate its viability. The controller architecture is that depicted in Fig.2 with L=8, M=2 and S=4.

The Fig.10(a) shows the controller output for a slow transitory bidimensional sweep (a D.C. analysis had a lot of convergence problems), where the singletons take the maximum and minimum values alternatively, thus showing clearly all the interpolation points.To illustrate the dynamic behavior, a wise transitory analysis makes the system to evolve through a trajectory that implies different scenarios in the surface depicted in Fig.10(a). The resulting transitories are shown in Fig.10(b) and Fig.10(c), where the curves correspond to static values of x_2 (from top to the medium level): 2.80V, 2.90V, 2.95V and 3V, while the input x_1 changes as indicated in the figure. The large overshootings that Fig.10(c) shows are due to changes in the boundary conditions of the destination point, i.e. in how it 'sees' the active rules (see Fig.1). In Fig.10(b) such neighborhood was similar for the initial and the target points, while it is different in Fig.10(c).

On the other hand, since the power of this strategy grows with the system complexity, let us make some simplifications to highlight such advantage over a fully analog implementation. Let us consider the area and power consumption of the interval selector and A/D converters negligible when compared to that consumed by the analog core and the memory (note that just one array of resistors is needed in the system, no mind how many converters are).

In addition, let us suppose a similar interconnection area (which indeed must be much smaller in the proposed strategy). Finally, since the digital memory has the same size in both implementations, the above assumptions allow to have an estimation for the area and power saving in the ratio between the total number of rules (L^M) which should be implemented without multiplexing and the number of active rules (2^M), which are those physically implemented in the proposal. Such ratio is,

$$\alpha = (L/2)^M \qquad (2)$$

which is strongly dependent on the number of inputs and labels per input, thus on system complexity. Hence, the higher the value of α in (2), the more suitable the proposed implementation is. Finally, remember that only the implemented rules, i.e. the active rules, contributes to the error at output, which allows the system expansion for a given error bound with respect to a fully analog implementation.

5. References

[1] K.A. Nishimura, "*Optimum Partitioning of Analog Mixed-Signal Circuits for Signal Processing*". Memorandum No. ECB/ERL M93/67, University of California at Berkeley. 1993

[2] P.P. Bonissone et al., "Industrial Applications of Fuzzy Logic at General Electric", *Proceedings of the IEEE,* Vol. 83, pp. 450-465, March 1995

[3] A. Costa, A. de Gloria, P. Faraboschi, A. Pagni and G. Rizzotto, "Hardware Solutions for Fuzzy Control". *Proceedings of the IEEE,* Vol. 83, pp. 422-434, March 1995.

[4] H. Eichfeld, M. Klimke, M. Menke, J. Nolles and T. Künemund, "A General-Purpose Fuzzy Inference Processor", *Proc. of the 4th Int. Conf. on Microelectronics for Neural Networks and Fuzzy System* , Sept. 1994

[5] F. Vidal-Verdú and A. Rodríguez-Vázquez, "CMOS Design of Analog Neuro-Fuzzy Controllers using Building Blocks" *IEEE Micro*, August 1995.

[6] T. Yamakawa, "A Fuzzy Inference Engine in Nonlinear Analog Mode and Its Application to a Fuzzy Logic Control". *IEEE Trans. on Neural Networks,* Vol. 4, pp. 496-522, May 1993.

[7] J.S.R. Jang and C.T. Sun, "Neuro-Fuzzy Modeling and Control". *Proceedings of the IEEE,* Vol. 83, pp. 378-406, March 1995.

[8] M. Brown and C. Harris, *NeuroFuzzy Adaptive Modeling and Control*, prentice Hall International, 1994.

[9] M. Masetti, E. Gandolfi, A. Gabrieli and F. Boschetti, "4 Input VLSI Fuzzy Chip Design able to process an Input Data Set every 320ns", *Proceedings of the JCIS'95,* Wrightsville Beach, North Carolina, USA, October 1995

[10] F. Vidal-Verdú, R. Navas and A. Rodríguez-Vázquez, "A Modular CMOS Analog Fuzzy Controller", *Proceedings of the FUZZ-IEEE'97*, Barcelona, Spain, July 1997.

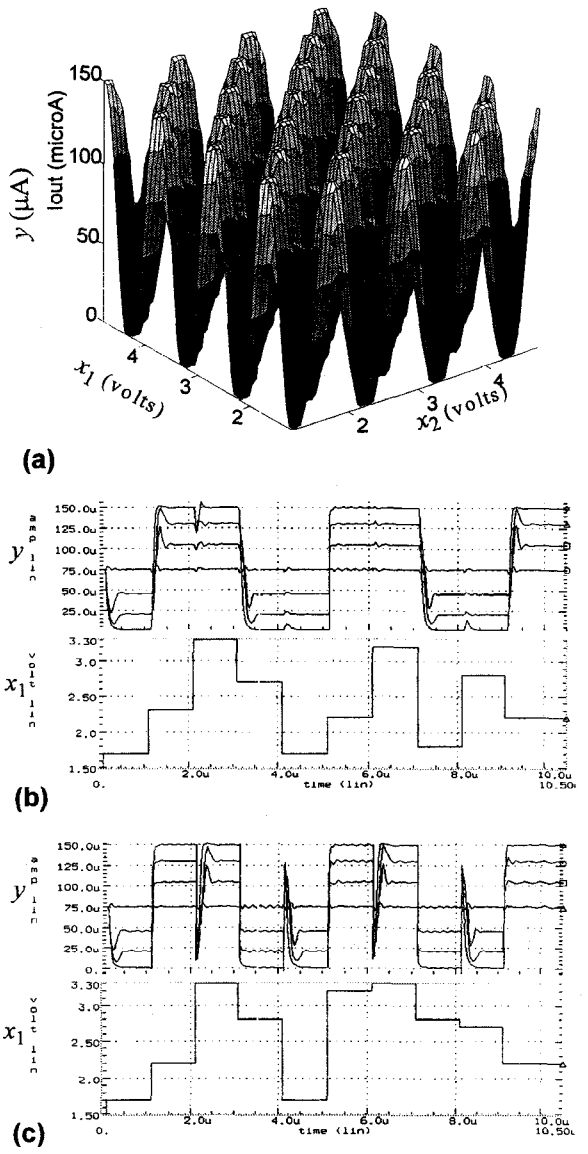

Fig. 10: Results from an Exemplary Controller; (a) Slow Transitory Bidimensiona Sweep, (b) y (c) Transitories for Differen Scenarios