Mathware & Soft Computing 6 (1999) 331-343

A Mixed-Signal Architecture for High Complexity CMOS Fuzzy Controllers

R. Navas-González**, F. Vidal-Verdú** and A. Rodríguez-Vázquez*
*Dpto. Analog and Mixed-Signal Circuit Design
Centro Nacional Microelectrónica. Univ. Sevilla
Edificio CICA, C/Tarfia s/n, 41012 Sevilla, SPAIN
angel@cnm.us.es
**Dpto. Electrónica Universidad de Málaga
Complejo Tecnológico, Campus de Teatinos, Málaga, SPAIN
rnavas@ctima.uma.es

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. An example 64-rule, 2-input, 4-bit singleton controller has been designed in a CMOS 0.7μmm technology to demonstrate the viability of the architecture. The measured input-output delay is around 500ns for a power consumption of 16mW and a chip area (without pads) of 2.65mm^2.

1 Introduction

The widespread interest in fuzzy logic technology motivates the implementation of special purpose ASICs (Application Specific Integrated Circuits) in the electronics design field, which speed up the computation of the fuzzy algorithm when required [2][3]. Such circuits can be digital [4][5] or analog [6][7][8]. 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 [8]. However, since the fuzzy algorithm involves global computation steps [9][10], 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 [10]. 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.

Masetti et al. 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 [5]. 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(a), for a bidimensional universe of discourse, any input pair (x_1, x_2) in the light shaded interval $_{Cij} = [(\epsilon_{1i}, \epsilon_{1i+1}), (\epsilon_{2j}, \epsilon_{2j+1})]$ maps into an output determined by the rules enclosed in the dark shaded interval, which are called 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 [10], 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}.

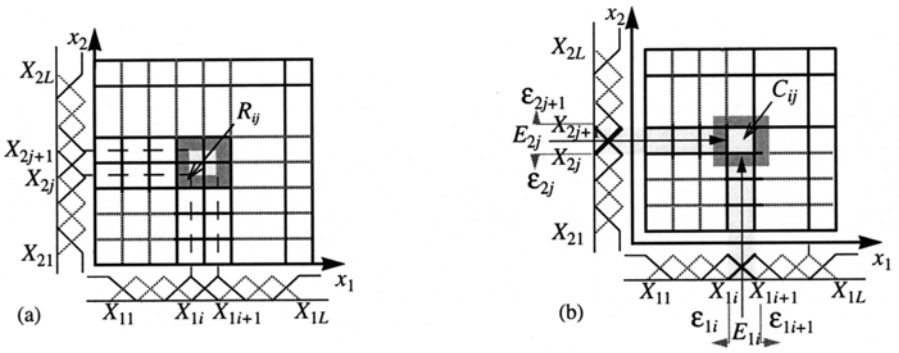

Figure 1. Illustration of the active rule set: (a) Lattice Partition; (b) Interpolation Intervals

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 $E_1, ...E_k, ...E_M$ 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 $y_1^*, ..., y_l^*, ...y_2^{M^*}$ 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

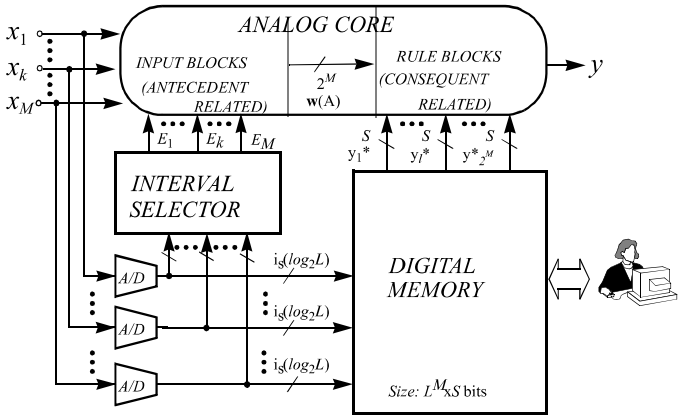

Figure 2. General Architecture of a Controller with Multiplexed Analog Core

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 [7], 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 [11], where the reader can find more details.

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) is 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 [11]. 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

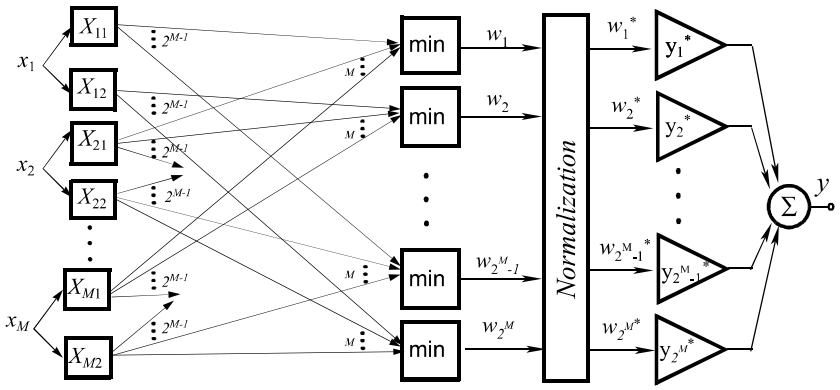

Figure 3. Analog Core Architecture

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 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 example interface to build the rule R_{ij} in Fig.1(a). In this example, we need two 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(b).

The high level block in Fig.4(c) is very similar to the rule block described in [7]. However, the singleton circuitry is slightly different. Since such cell is set up dynamically by the digital word $s_{l0}...s_{lS-1}$, 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 $\bar{s}_{l0}...\bar{s}_{lS-1}$. 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, $y = \sum_{l=1...2^l} y_l$ which is realized just by attaching the rule block output nodes, because the rule block outputs are currents.

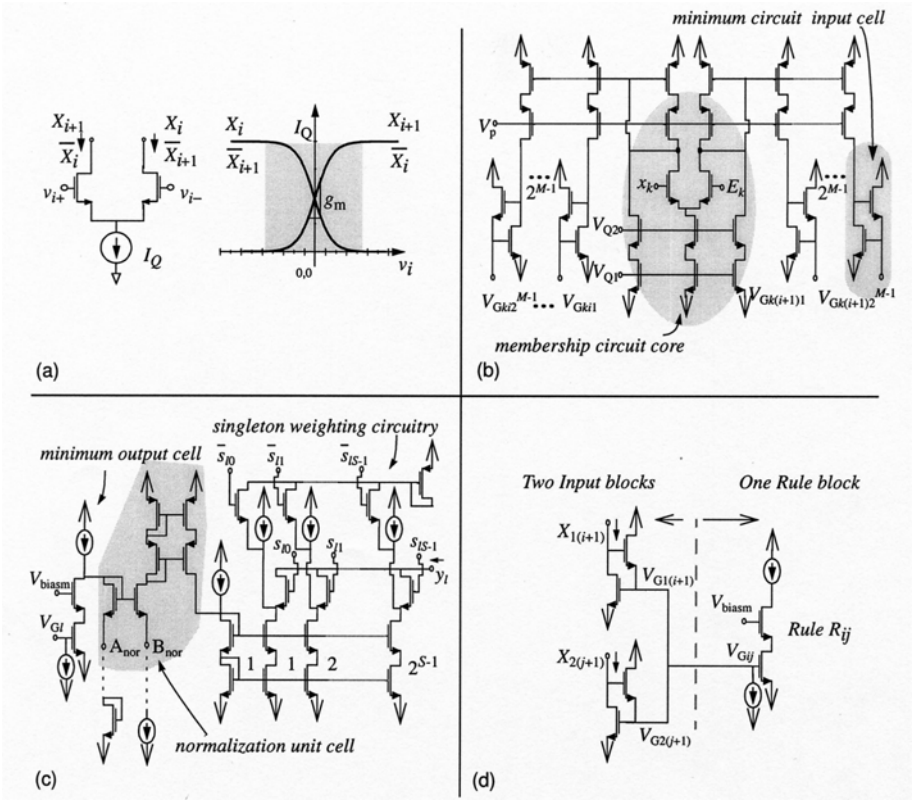

Figure 4. Analog Core Implementation: (a) Membership Function Generation, (b) Input Block and c) Rule Block (d) Interface

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(a) 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

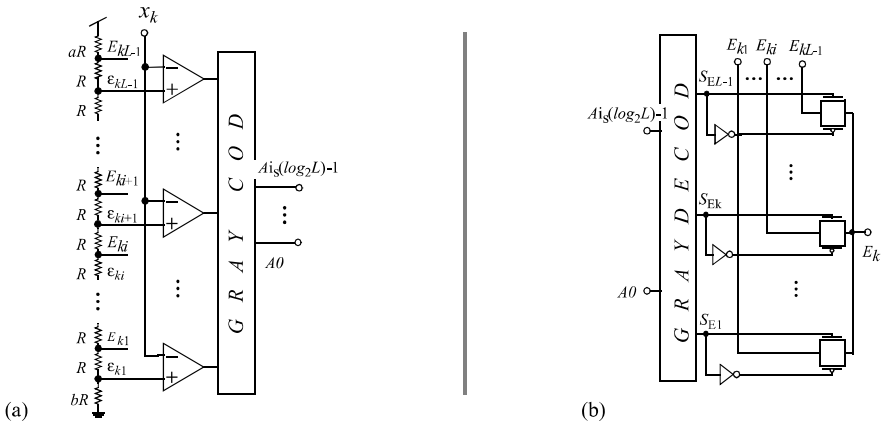

Figure 5: (a) A/D Flash Converter ; (b) Interval Selector k Cell

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.5(a), $E_1 ... E_M$. The Fig.5(b) 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.

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 $2^M \times S$ 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 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.6(a),

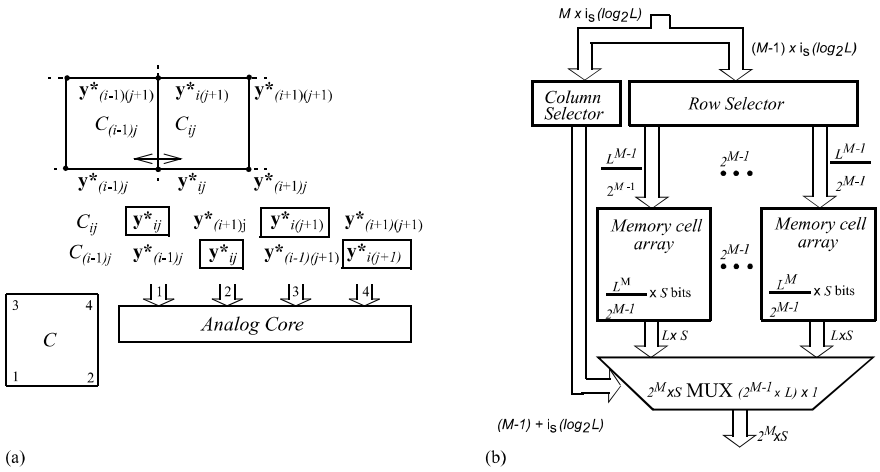

Figure 6: Digital Memory: (a) Analog Core Memory Interface; (b) Digital Memory architecture

The Fig.6(b) shows the proposed memory architecture that follows the latter approach. Singleton values are distributed into memory cell arrays of rows and L columns. Every memory cell array stores pairs of singleton values which are needed to be addressed simultaneously. The row selector selects rows simultaneously, one for each memory cell array, to provided a set of 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 2M x S output word, all in one step. To illustrate the data distribution and memory operation let us show the situation for a bidimensional case. The Fig.7(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.7(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.6(b) the result of a row selection is shown for the four intervals in Fig.7(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

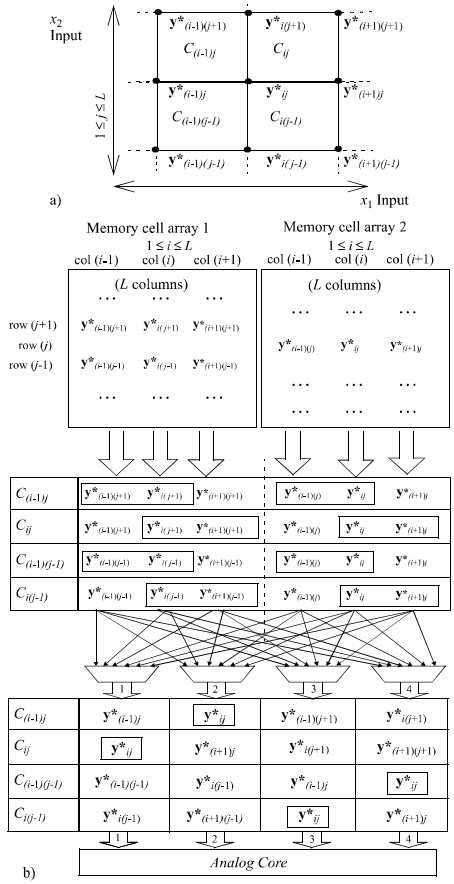

Figure 7. Memory operation example: (a) Four generic intervals; (b) Singleton read example.

4 Results and conclusions

An example 64-rule, 2-input, 4-bit per singleton controller (, and) has been designed in a CMOS 0.7mm technology to demonstrate the viability of the multiplexed architecture in Fig.2. Fig.8(a) shows a microphotograph of the chip, where the die area is 5mm2. However, the latter is determined by restrictions of the foundry, while the actual chip area consumption is smaller. Specifically, the summation of the building blocks area consumption is below 1mm^2. Fig.8(b) shows

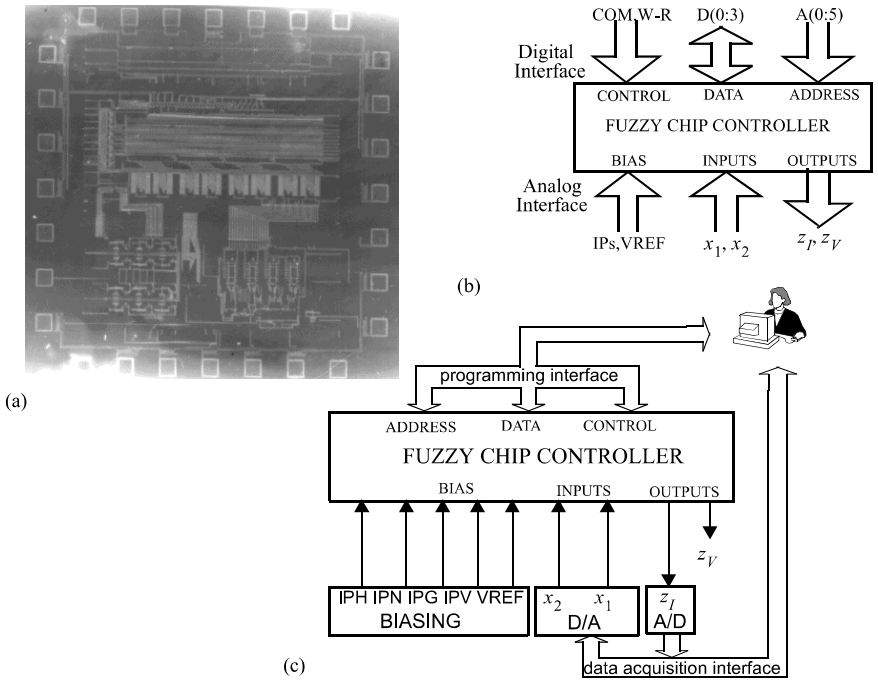

Figure 8. Microphotography of the chip (a); chip interface (b) and testing board block diagram (c).

Fig.9 shows some experimental results obtained from this test environment. Fig.9(a) and Fig.9(b) depict the section of an example dc control surface (state-steady response) and the transient response to a step input respectively, both measured with the oscilloscope. The former illustrates the response (bottom) to a slow ramp in one input (top) while the other input remains constant, in a kind of mexican hat control surface. The latter corresponds to a falling edge in one input that forces the output to change from its maximum to its minimum value, as well as to jump to a different interpolation interval, which means a dynamic programming of the analog core. The measured delay time is around 500ns. Since the oscilloscope is not able to sense currents, previous measurements are voltages in the output . With regard to Fig.9(c) and Fig.9(d), they are built with data from the data acquisition board in Fig.8(c), where the current output of the chip is externally converted into a digital word and processed. These two examples illustrate the ability of the controller to interpolate nonlinear (Fig.9(c)) as well as linear control functions (Fig.9(d)).

On the other hand, the measured chip power consumption was around 16mW, which is obtained by sensing the current from the supply voltage source. The controller was designed to take into account mismatching among transistors for a standard deviation (s) associated to the output that equals an error of 3.3%, while the measured errors were around 10%. The latter corresponds to a worst case and still fit most control requirements [8].

It is illustrative to compare the speed, area and power of this mixed-signal controller to that of a pure analog one.While the circuit in [7] (designed in a 1mm technology) comprises 16rules with 470ns delay, 8.6mW power consumption and 1.6mm2 area, the presented controller implements 64rules with almost the same delay, 16mW power and only 2.6mm2 area.

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). 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, $\alpha = (L/2)^M$ which is strongly dependent on the number of inputs and labels per input, thus on system complexity. Hence, the higher the value of a, 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

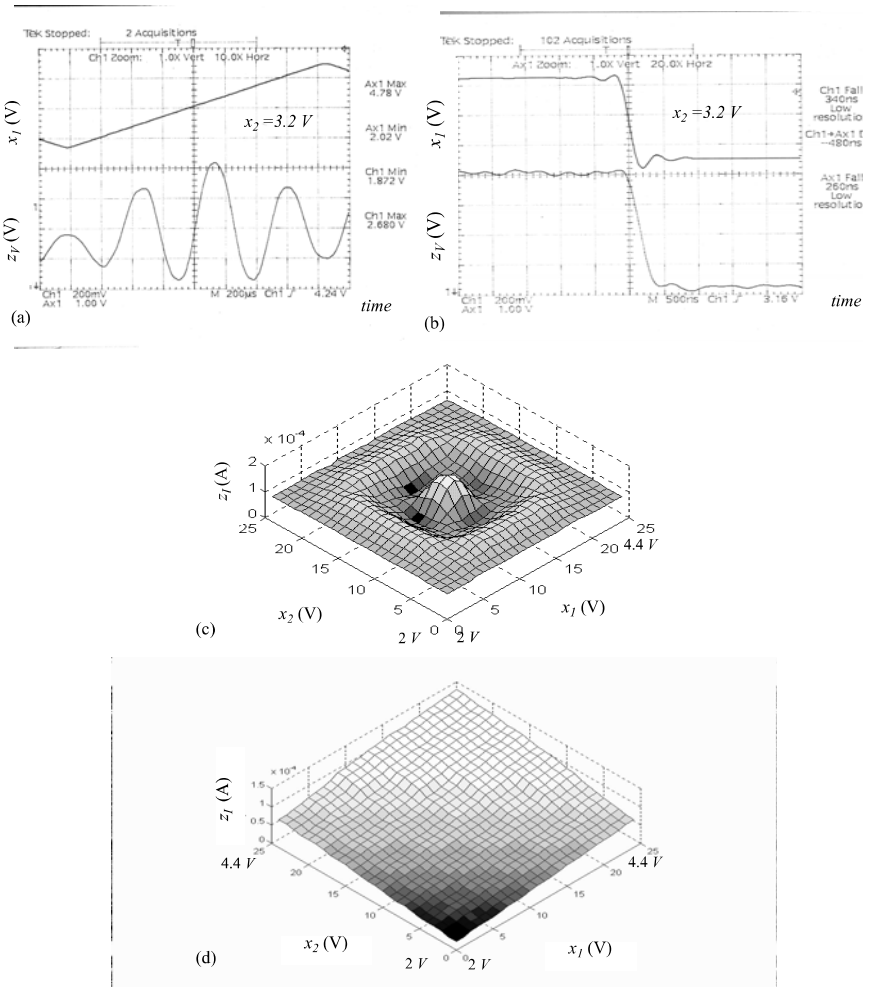

Figure 9. Measured results: (a) DC nonlinear control surface section, b) transient response, (c) nonlinear control surface and (d) linear control surface.

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] 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

[6] S. Guo, L. Peters, and H. Surmann, "Design and Application of an Analog Fuzzy Logic Controller". *IEEE Trans. on Fuzzy Systems*, Vol. 4, pp. 429-438, November 1996.

[7] A. Rodríguez-Vázquez, R. Navas, M. Delgado-Restituto and F. Vidal-Verdú, "A Modular Programmable CMOS Analog Fuzzy Controller Chip", *IEEE Transactions on Circuits and Systems II*, Vol. 46, pp. 251-265, March 1999.

[8] 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.

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

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

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