A Mixed-Signal Fuzzy Controller Architecture

Fernando Vidal-Verdú **, Rafael Navas-González **, and Angel Rodríguez-Vázquez *

*Dept. of Analog and Mixed-Signal Circuit Design
Centro Nacional de Microelectrónica-Universidad de Sevilla Edificio CICA, C/Tarfia s/n, 41012-Sevilla, SPAIN
FAX:: 34 5 4624506 Phone:: 34 5 3623811 email: angel@cnm.us.es

**Dto. de Electrónica
Universidad de MálagaComplejo Tecnológico, Campus de Teatinos, Málaga, SPAIN
FAX: 34 5 2132781 Phone: 34 5 2133325 email: vidal@ctima.uma.es

Abstract

Limits to precision impose limits to the complexity of analog circuits, 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 marked increment in 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. HSPICE simulations from an example controller are shown to illustrate the viability of the proposal.

1. Introduction

Because analog fuzzy controllers operate in fully *parallel* mode, they are well suited for applications that require high operation speed. However, the most complex *monolithic* analog controller chips reported to date feature 13-rule at 3-input [1] and 16-rule at 2-input [2] – much smaller than the up to 100-rule and 4-input reported for digital chips [3].

A major obstacle to increasing the complexity of fully-analog controller chips is the existence of *global* computation nodes where the errors caused by the rule antecedent circuits are aggregated; for instance, the center of gravity is evaluated in such a node. This obstacle is specially pertinent for those architectures based on *lattice* partitions of the universe of discourse because these partitions (otherwise advantageous for control applications [4]) make the rule count increase exponentially with the input count. Thus, the accumulated error increases exponentially as well, and the system architecture may become unfeasible even for low numbers of inputs.

This paper presents a novel *mixed-signal* controller chip architecture which maintains the fully-parallel operation feature of analog chips but obtains the accumulated error independent of the number of rules. It takes advantage of the fact that each fuzzy rule influences the system output only inside a *local*, limited region of the universe of discourse. Consequently, the universe of discourse can be split into sub-regions and the output for each sub-region can be calculated by only evaluating only a reduced number of fuzzy rules (called *active* rules [5]). This number is the same for each sub-region, but the corresponding rule parameters differ from one region to another. Our proposal implements this reduced number of rules by a low-dimension *programmable* analog processor, and employs *multiplexing* to cover the entire universe of discourse. On chip digital circuits are employed for multiplexing and for programmability.

2. Architecture and Functional Description

Consider for illustration purposes the bi-dimensional lattice partition of Fig.1. It shows the universe of discourse split into *interpolation* intervals, each having a different set of active rules. For instance, any input pair (x_1, x_2) in the light-shaded interval $C_{ij} = [(\varepsilon_{1i}, \varepsilon_{1i+1}) \times (\varepsilon_{2i}, \varepsilon_{2i+1})]$ maps onto an output determined by the rules in the dark shaded interval (active rules) while all remaining rules have no influence on the output. Only the active rule membership functions and their associated singleton values are needed for the interpolation procedure. In addition, the only membership function pieces needed are those which actually contribute to the system output. Thus, in the case illustrated in Fig.1, only the pieces drawn with thick lines and the singletons associated with the four active rule consequents, y^*_{ij}, $y^*_{i(j+1)}$, $y^*_{(i+1)j}$ and $y^*_{(i+1)(j+1)}$ are needed to generate the output in the interval C_{ij}.

Fig.2 shows the proposed architecture for a controller with M inputs, L labels per input (thus L^M rules), and with S bits per singleton. It comprises the following blocks:

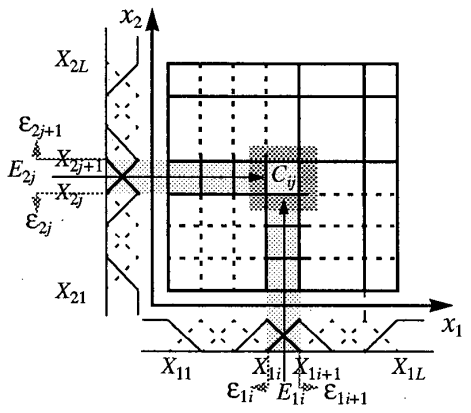

Fig. 1: Illustration of the active rule set: (a) Lattice Partition; (b) Interpolation Intervals

0-7803-5008-1/98/$10.00©1998 IEEE.

- *A/D Converters*: Their function is to encode the interpolation interval C_{ij} associated with the current input. There are M, one per input, with a resolution equal to the next superior integer $\log_2 L$, i.e. $\text{int}_s(\log_2 L)$. Thus, this battery of converters provides a word of $M[\text{int}_s(\log_2 L)]$ bits that drives the *interval selector* block and the *digital memory* block.
- *Interval Selector*: It selects a set of voltage values $E_1 \ldots, E_k \ldots E_M$ to drive the analog core and, thus, makes it implement the active membership functions.
- *Digital Memory*: It selects the active 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.
- *Analog Core*: It performs the fuzzy computation having 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 specifying the membership functions associated with the rule antecedents as well as the singleton values related to the consequents.

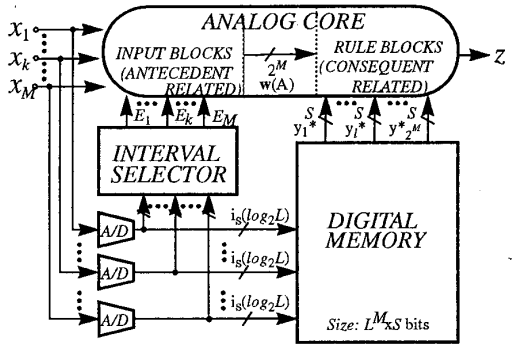

Fig. 2: General Architecture of a Controller with Multiplexed Analog Core.

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 [2], 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 [2], where the reader can find more details.

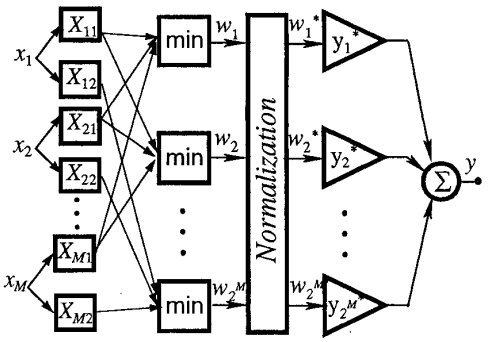

Fig. 3: Analog Core Architecture

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 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. In addition, 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 [4]. 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 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. 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

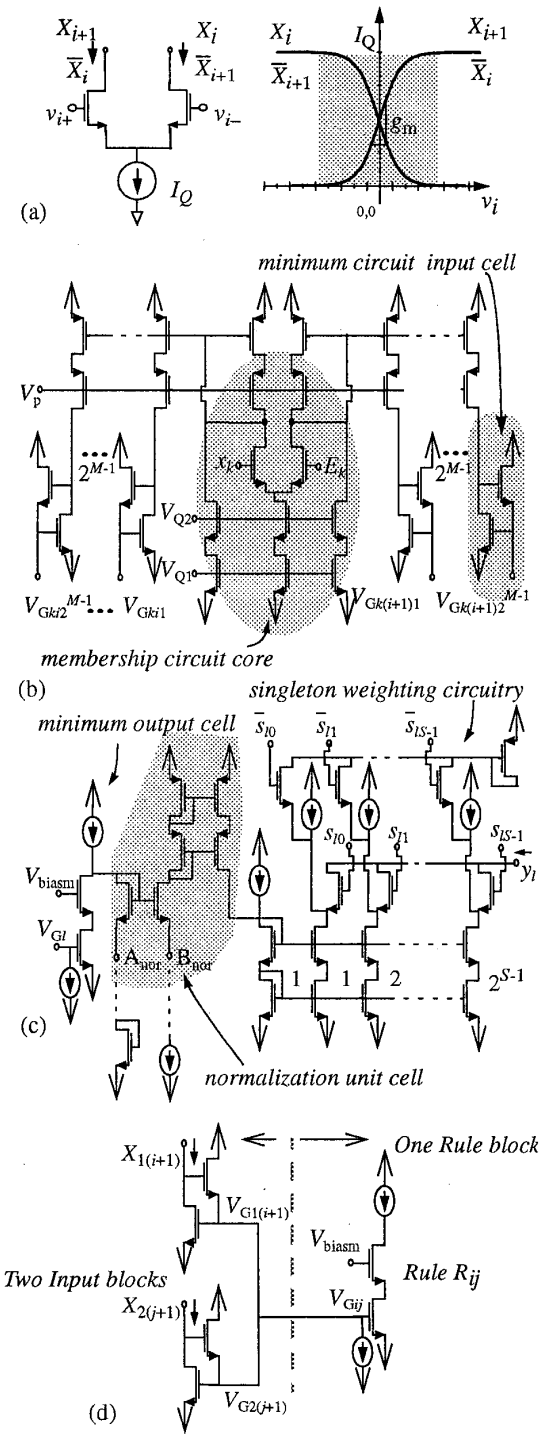

Fig. 4: Analog Core Implementation; (a) Membership Function Generation, (b) Input Block and (c) Rule Block (d) Interface

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 [2]. The system global output is obtained by aggregating the rule blocks outputs, $y = \sum_{l=1\ldots 2^L} y_l$ 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(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. 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.

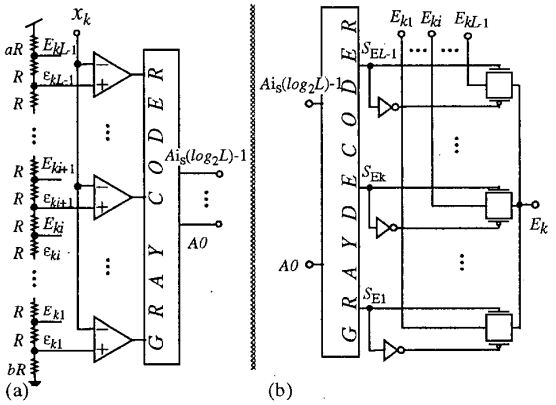

Fig. 5: (a) A/D Flash Converter; (b) Interval Selector k Cell

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), $E_1 \ldots E_M$. 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 $Mlog_2L$ provided by the A/D converters and associated to the system inputs is used to address a digital memory. Such memory stores the singleton values in a digital code. The digital memory must supply a word of $S \times 2^M$ bits, where S is the number of bits per singleton value. However, since each singleton value is used by all the adjacent intervals, special care is needed to design the memory architecture. A possible solution consists in generating one word with the singleton values of the active rules for each interval, and using a conventional memory. However, this implies to replicate each singleton value as many times as adjacent intervals it has, which is 2^M. A second solution multiplexes the memory output bus and organizes data to avoid redundancy, which is the strategy followed by the example controller in this paper. This multiplexing is not made in time, data is put in the output bus in one step.

4. Results and conclusions

An example 64-rule, 2-input, 4-bit singleton controller ($L = 8$, $M = 2$ and $S = 4$) has been designed in a CMOS 0.7μm technology to demonstrate the viability of the proposed architecture.

High-speed flash A/D converters are used for interval encoding. The poly-silicon resistor array used to generate the converter thresholds is also exploited to generate the interval selector voltages. Thus, this latter block includes only analog switches and some logic. The memory block is designed carefully so that the data are put in the singleton programming bus in the proper order – necessary to reduce the memory size by a factor of 4. Finally, the 4-rule 2-input processing core is constructed by using building blocks similar to those presented in [2].

Fig.6(a) shows the DC surface response of the controller for a case where the singletons are chosen to clearly display all the interpolation points. The DC accuracy was around 1%. Fig.6(b) illustrates controller transient behavior for different values of x_2 (2.80V, 2.90V, 2.95V and 3V) forcing x_1 to sweep for each of these values. The input was chosen so that the trajectory goes through different sub-regions. The delay time, including the time required for dynamic reconfiguration, is around 500ns. Chip power consumption is 16mW.

It is illustrative to compare the speed, area and power of this mixed-signal controller to that of a pure analog one with the same number of inputs and rules and digitally-programmable singletons. Assuming that the singletons are encoded with the same number of bits, the size of the digital memory is the same for the two controllers. However, while the number of rule blocks is L^M for the fully-analog one, it is only 2^M for the new one. The larger the ratio $\alpha = (L/2)^M$ the more advantageous and less error-prone is the new architecture as compared to a fully-analog one. Thus, while the circuit in [2] (designed in a 1mm technology) comprises 16rules with

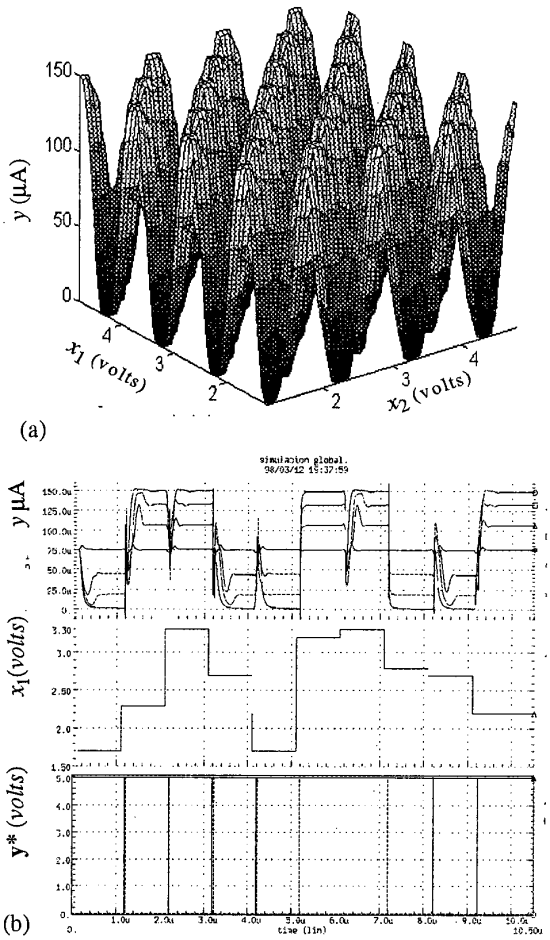

Fig. 6: Results from an Example Controller; (a) DC surface response, (b) transient response behavior

470ns delay, 8.6mW power consumption and 1.6mm2 area, the example controller implements 64rules with almost the same delay, 16mW power and only 1mm2 area

References

[1] 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.

[2] F. Vidal-Verdú, R. Navas and A. Rodríguez-Vázquez, "A 16 Rules@2.5Mflips Mixed-Signal Programmable Fuzzy Controller CMOS-1μm Chip", *Proc. of the ESSCIRC'96*, pp. 156-159, Sept. 1996.

[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] M. Brown and C. Harris, *Neuro-Fuzzy Adaptive Modeling and Control*. Englewood Cliffs: Prentice Hall 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.