Mathware & Soft Computing 3 (1994) 297-308

Systolic Architectures for Fuzzy Processing and their Simulation

Luis de Salvador[1], Marcos García[1], Julio Gutiérrez[1,2]
[1] Laboratorio de Hardware y Control Avanzado
Inst. Nac. de Técnica Aeroespacial. Ctra. Ajalvir km. 4.
28850 Torrejón de Ardoz. Madrid. España
e-mail: salvadorla@inta.es
[2] Dept. Tecnología Fotónica. Univ. Politécnica de Madrid
Campus de Montegancedo. 28660 Madrid

Abstract

This paper details the study of systolic architectures for fuzzy rules processing made at the Hardware and Advanced Control Laboratory - INTA. The theoretical basis of these architectures is described and analysed. Likewise, the resultant schematics are simulated using a hardware description language (VHDL) with standard cells from ES2. This gives us a very accurate assessment of their real performance. In this way we can detect the inherent shortcomings in this class of systems and we outline several ways of overcoming then.

Keywords: Fuzzy, Fuzzy Control, Systolic, Simulation, VHDL.

1 Introduction

One of the fields of research in fuzzy logic is the development of specific hardware with which we can obtain all the advantages of working with this class of models. There are, basically, two areas of research in fuzzy controllers: analog design and digital design. The latter brings out solutions based on systolic architectures. To systolize a problem consists in dividing it into identical elementary operations that can be executed together synchronously in parallel [1,2]. We can represent all these elementary operations as a set of homogeneous processing units that interconnect together giving an array of n dimensions. Depending on the design of this array, the inputs and outputs are made through all or a subset of units which are in the limits of the array. The theoretical basis of systolization of fuzzy inference processing defined by several authors [3,4,5,6]. These bases have lead to the design of several systolic architectures for processing. At this stage it is convenient to use simulation tools. These tools allow:

1.- The formalization of design specifications, so that it will also formalize the theory upon they are based.

2.- The study and checking of the designs before their implementation.

3.- The testing of system performance so as to check failures in design theory.

The simulation tool we use is a VHDL1 simulator. The adequacy of VHDL as a specification and simulation language is justified by its own definition [7,8,9,10].

2 Theoretical Basis

In this section we present the basis of systolic processors of fuzzy rules. By default, we consider the basis of fuzzy logic already known [11,12].

To develop a systolic architecture, most authors [3,4,5,6] follow several steps, as described below:

1. Definition of a set of rules of the model to be developed. These rules are as follows:
   IF A_1 is $LBA_{1,x}$ and/or .... A_n is $LBA_{n,x}$
   THEN B_1 is $LBB_{1,y}$ and/or B_m is $LBB_{m,y}$
   where n is the number of antecedents, $LBA_{a,b}$ the b-th label of the antecedent variable A_a, m the number of consequents and $LBB_{a,b}$ the b-th label of the consequent variable B_a. They comply with the condition $\{A_x\} \cap \{B_y\}$ is empty, that is to say, the rule system is not linked. The key words "and" and "or" are, respectively, the norma and co- norma used [11,12].

2. Modification of the rule set, so that each of them involves only one consequent variable. The rules will have the form:
   IF A_1 is $LBA_{1,x}$ and/or .... A_n is $LBA_{n,x}$
   THEN B_a is $LBB_{a,b}$

3. Normalization of the antecedent side to eliminate the "or" operators and to expand the rule, so that all the antecedent variables will be included there. This, as we will demonstrate later, is a great mistake from a theoretical point of view, giving a rule set that is not equivalent to the initial one.

4. Including in one set all the rules that have the same consequent variable.

   From now on, the rest of the process is described in one set, performing the same operations for each different set.

5. Creation of a "rule basis". This rule basis is a matrix that contains all the information about conditions to be kept by the antecedents in order that variable B_a will have one result or other. The creation of the matrix is done in the following way:

5.1. The dimension will be the number of antecedents plus one $(n+1)$.

5.2. The length of each dimension will be the number of labels of the associated variable.

5.3. The position $(x_1, x_2, ..., x_n, x_{n+1})$ will take on the value depending on whether the rule
IF A_1 is LBA_{1,x_1} and ... A_n is LBA_{n,x_n}
THEN B_x is $LBB_{x,x_{n+1}}$
belongs to the set of rules or not. It is also possible to consider values between 0 and 1, to consider the fuzzification of antecedent conjunction or the insertion of modifiers.

Note point 5.2, which shows one important conceptual change in relation to other authors [3,4,5,6], who consider the length of each dimension as the size of the universe of discourse of the associated variable. After several studies undertaken in this laboratory, we have seen that the use of only the number of labels is sufficient if at the data entry stage we introduce one fuzzifier unit. This reduces the size of the rule basis by two orders of magnitude, making it easier to use, and also substantially reduces the number of operations to be executed, so increasing the performance of the designed architecture. In the way that we designed the architecture, the fuzzifier unit does not generate any delay which might result in a reduction of process speed, so that the gain obtained using this technique is high.

6. Generation of input data matrix. This matrix is generated as follows:

   6.1. The dimension will be the number of antecedent variables.

   6.2. The length of each dimension will be the number of labels of the associated variable.

   6.3. The position $(x_1, x_2, ..., x_n)$ will have the smallest membership value of data inputs to labels LBA_{1,x_1} ... LBA_{n,x_n}.

7. Multiplication of both matrices to obtain the inference result. This step is the one really systolized, and it is possible to do it in several ways.

8. Defuzzification of result. This step could be performed using any of the existing methods.

2.1 Systolization

In our model, instead of multiplying two matrices, as described earlier, we reduce the rule basis matrix in one dimension, generating several matrices with the information of only one label of each consequent variable. In this way, the result of the matrix multiplication gives the membership of the consequent variable to this label.

Figure 1: Systolization scheme

To compute the multiplication we will use one max-min unit, which will be described later, requiring as many units as the number of labels of consequent variables.

The diagram that describes the behaviour of the systolic system is shown in Figure 1.

This figure shows the general behaviour of the systolic system described above. In summary, the set of possible rules is evaluated sequentially. The result is inserted in a "minimum" unit (represented by X) simultaneously with a binary vector. This vector complies with rules that have an effect on each consequent label. Once all of the rules are evaluated, the values obtained are defuzzified.

3 Architecture

The architecture described in the previous section is comprised of three main components:

1. The fuzzifier
2. The fuzzy inference unit
3. The defuzzifier

The system, at this point, appears as a pipeline of three segments. This means that the running of these segments is fully synchronised: the maximum delay in any of them determines the performance of the system.

In this case, system perfomance is limited by the intermediate unit: the fuzzy inference unit.

Figure 2: Structure max-min inference

3.1 Fuzzifier

In the architecture developed, the fuzzifier has been implemented by means of memory banks: one memory for each antecedent variable. The entry point to the memory located by an antecedent variable is an index whichs gives the correspondence between the external universe value and the universe of discourse: a value of 8 bits in a universe of discourse of 256 locations. Each entry point references one word of length equal to the number of possible labels for the antecedent, multiplied by the precision in bits used in the evaluation of the membership function:

Entry_ x: [label 1][label 2]...[label 7]

In the simulation of this design we use up to 7 labels with three precision bits in the membership function. In this way, the shape of the membership function we can configure is free, without any constraints. Hence, each memory has as many outputs as labels that have been defined in the universe of discourse.

3.2 Inference unit

The fuzzy inference unit developed here is in accordance with guidelines described by other authors [3,4,5,6], but with one exception that is essential in our architecture. This is the representation of the rules map as indicated in section 2. We divide the inference unit into three main parts:

a. Systolic array

b. Multiplication of antecedents

c. Rules map

The systolic array is formed by a chain of "max-min" units, as shown in Figure 2.

Figure 3: Max-min unit detail

Each "max-min" unit corresponds to one elementary operator as described in the previous section, with the following optimization: we carry out the minimum from the logical product (logical AND) of antecedents. This part is common to all "max-min" units. The membership function is encoded by three bits but, is later decodified at 8 bits (unary codification), which allows the evaluation of maxima and minima using only one level of gates (Figure 3) and subsequently the building of high speed units.

The antecedent product unit is made up of a set of shift registers. The length of each shift register is equal to the maximum number of labels that an antencedent variable contains. This number of antecedents is set at synthesis time. The output dimension of the shift registers is the equal to the number of bits used to encode the membership function. In our case, it is three bits wide. An auxiliary counter circuit manages the output of antecedent labels, and optimizes the circuit perfomance when the number of antecedents used in the definition of the fuzzy system is less than the maximum number of antecedents available in the hardware system.

The rule map is made up of interleaved memory banks. Each memory bank corresponds to a consequent variable and a label of the same consequent. It must provide the membership encoding flow (1's and 0's data flow) at the same speed that the "max-min" units work. These units process so fast that it is necessary to build a pipeline structure that simulates a memory with the required latency.

3.3 Defuzzifier

The defuzzifier unit works at the same speed that the inference hardware system generates final values. The method used to process defuzzified values is centroid computation. This units contains:

1. Parallel multiplier unit [1,2].
2. Built-in-memory divider.

Figure 4: Systolic architecture scheme

We can use a built-in-memory divider since we only use eight different levels to encode the membership values. This procedure allows us to develop a simple and very fast design. Moreover, the through-put of the defuzzifier is equal to the through-put of the fuzzifier.

The whole architecture scheme of the systolic system is depicted in Figure 4.

4 Simulation

Between the design specification and its actual implementation there exists an essential step: simulation using CAD tools. The main problems and disfunction hidden in the architecture emerge during this phase. Moreover, we can set the most important parameters of perfomance such as: estimated time to process, through-put, and so on.

We have employed a hardware description language, VHDL[1], as a simulation tool. This approach has many advantages: it is a standard tool [7,8], it permits to design with modularity criteria, it is able to synthesize over standard cells or FPGA's and we can build parametrizable designs. This last feature allows us to define many parameters of the circuit in a flexible way. In our case, the set of parameters is:

1. Number of antecedents.
2. Number of consequents.
3. Labels per antecedent.

4. Labels per consequent.

5. Maximum number of labels per antecedent.

The libraries have been developed in our laboratory with the ES2[2] specifications for $1\mu m$ cells .

Because the execution time of the inference unit depends on the definition of the number of labels per antecedent, the cicuit has been adapted in order to be able to set the right number of labels per antecedent used. In this way the best perfomance for every configuration of the fuzzy system is achieved.

Simulation was performed over a circuit generated with the following parameter settings:

1. Four antecedents with a maximum number of seven labels.

2. Seven labels per consequents.

3. Universe of discourse with 256 samples.

4. Three bits to encode the membership function.

5. Up to 8 Kbytes of memory.

This circuit was programmed to hold four antecedents and three labels per antecedent. The perfomance of the circuit is the following:

1. The circuit can be run with a clock period of 10 ns.

2. The whole rule base is executed in only 810 ns.

3. The through-put of the system is 1.2 MFLIPS[3].

4. The rule execution frequency is 100 MHz.

5. The stimated silicon area is 40-60 mm^2.

6. The amount of time necesary to program the rule base and fuzzifier is 46.4 ns.

This architecture therefore provides very high perfomance in relation to other developments, even non-systolic designs [13,14,15], especially if we take into account that this is not the maximum perfomance of the system. Moreover, this design does not need much memory, unlike early fuzzy processign systolic architectures. Finally, the silicon area is inside the bounds of a small dice of silicon.

5 Issues that Arise in Systolic Designs

In developing the present architecture we have detected some conceptual disfunctions and several critical aspects of the simulation process. These issues are described in the following subsections: theoretical issues (the most important ones) and implementation issues.

5.1 Theoretical Issues

One of the advantageous characteristics of fuzzy systolic architectures is their presummed their capability of processing all the possible rules of the fuzzy system [3,4,5,6].

Although this aspect seems obvious for several authors[4], the following problem arises:

- Consider a control system with two input variables [A, B] and one output variable [C].
- Impose the condition of only two labels per variable [LBA_1 LBA_2], [LBB_1 LBB_2], [LBC_1 LBC_2].
- Assume a fuzzy systolic design.

In the presented case, it is not possible to represent in the fuzzy systolic system the following rule:

IF A is LBA_y THEN C is LBC_y

Because the rule system is represented orthogonally, the most aproximate rule we can use is:

IF A is LBA_y and (B is LBB_1 or B is LBB_2) THEN C is LBC_y

Althoug both sentences are equivalent in classic (or crisp) logic, this is not so in fuzzy logic. The reason is that there is not a complementary property in fuzzy logic [11]. This means:

$$u_{LBB1}(x) \text{ or } u_{LBB2}(x) \text{ does not imply } 1$$
$$u_{LBB1}(x) \text{ and } u_{LBB2(x)} \text{ does not imply } 0$$

A very similar result occurs in a rule where the negation of an antecedent appears.

Our conclusion is the following: fuzzy systolic systems, both the traditional developments and the system depicted in this paper, process only a limited set of rules. These rules have an antecedent part made up of the logic product of some label of every antecedent defined in the fuzzy system.

5.2 Implementation Issues

Among the critical issues that arise in the design and simulation of systolic systems at implementation, we stress the following:

1. Required memory capacity grows exponentially with the number of antecedents that the system holds. This problem is a limitation of these systems, since it is not possible to build a circuit with a high number of antecedents due to the limits of dice size.

2. The process speed is highly dependent on the number of antecedents and the number of labels per antecedent.

3. These kind of fuzzy architectures require the processing of a large number of rules. This is in conflict with the principles of fuzzy modelling, since one of its prime objectives is economy of rules.

4. If we use "max-min" units which use membership binary encoded values, this would result in a slow inference unit.

5. If we make use of "max-min" units with encoded membership values unity, the result is fast enough, but we must use many lines of data to encode several logical levels.

Finally, it must be kept in mind that all of these issues only affect very complex fuzzy control systems, that is, fuzzy systems with a high number of antecedents and labels per antecedent. Actually, fuzzy systolic architectures are systems suitable for current applications. There are choices that allow high perfomance without great cost in terms of silicon area.

Conclusions

The architecture explained in this paper shows two fundamental aspects of the systolic architectures for fuzzy processing.

First, the optimization that can be achieved when we employ the membership value product of every antecedent variable instead of the universe of discourse product of every antecedent variable. This modification puts fuzzy systolic architectures in a competitive position in terms of perfomance level. This implies an important silicon area saving if we compare this design with early developments.

Second, we have found a power process limitation in fuzzy systolic architectures. Fuzzy systolic systems do not execute all the possible rules of the fuzzy system. On the contrary, it is only possible to execute a subset of the rules of the fuzzy system: those that have an antecedent part made up of the logic product of some label of every antecedent defined in the fuzzy system. This is because fuzzy logic does not have the complementary property that applies in crisp logic.

At this time, a great deal of development effort in the definition of this kind of system remains to be done but, even now, it is possible to employ this architecture for control systems. Implementations must be oriented to the expansion of the set of rules used to compute the inference values and to improving the relation between through-put and the number of antecedents. Currently, these tasks are

being carried out in the Laboratory of Hardware and Advanced Control in the National Institute of Aerospace Technology.

Notes

(1) V/System of Model Technology

(2) European Silicon Structures, library with industrial specifications.

(3) Fuzzy Logic Inferences Per Second.

(4) This is not justified by any author in a formal way.

References

[1] Hwang, Kai. *Computers Arithmetic.* John Wiley & Sons

[2] Hwang, Kai. *Advanced Computer Architectures.* McGraw-Hill

[3] M.A. Manzoul and S. Tayal. Systolic VLSI Array for Muti-variable Fuzzy Control Systems. *Cybernetics and Systems: An International Journal*, 21

[4] M.A. Manzoul. Fuzzy Inference on a Systolic Array. *Proceedings of 18th Modeling and Simulation Conference.*

[5] F. Fernandez, A. Ruiz, J. Gutierrez. Diseño Sistemático de un Procesador Sistólico de Inferencias Difusas. *II Congreso Español sobre Tecnologías y Lógica Fuzzy*

[6] Bugarín, A. y Barro, S. y Ruiz, R. Soluciones Sistólicas en Control Borroso. *II Congreso Español sobre Tecnologías y Lógica Fuzzy*

[7] IEEE Std 1076-1987. *IEEE Standard VHDL Language Reference Manual.* IEEE Inc.

[8] IEEE Standards Interpretations. *IEEE Standard VHDL Language Reference Manual.* IEEE Inc.

[9] Navabi, Z. *VHDL, Analysis and Modeling of Digital Systems.* McGraw Hill

[10] Coelho, D. *The VHDL handbook.*

[11] G.J. Klir and T. Foller. *Fuzzy Sets, Uncertainty and Information.* Prentice-Hall

[12] B. Kosko. *Neural Networks and Fuzzy Systems.* Prentice- Hall

[13] H. J. Zimmermann. *Fuzzy Technologien. Prinzipien, Werkzeuge, Potentiale.* VDI Verlag.

[14] M. Sasaki, F. Ueno, T. Inoe. 7.5MFlips Fuzzy Microprocessor Using SIMD and Logic-in-Memory Structure. *2nd IEEE International Conference on Fuzzy Systems.* 1993

[15] V. Catania, A. Puliafito, M. Russo, L. Vita. A VLSI Fuzzy Inference Processor Based on a Discrete Analog Approach. *IEEE Transactions on Fuzzy Systems.* May 1994