# WARP:
# WEIGHT ASSOCIATIVE RULE PROCESSOR
# A DEDICATED VLSI FUZZY LOGIC MEGACELL

A. Pagni, R. Poluzzi, G.G. Rizzotto

Corporate Advanced System Architectures
SGS-THOMSON Microelectronics
Via C. Olivetti 2
20041 Agrate Brianza (MI) ITALY

*Abstract.*

*During the last five years Fuzzy Logic has gained enormous popularity, both in the academic and industrial worlds, breaking up the traditional resistance against changes thanks to its innovative approach to tackling problems.*
*The success of this new methodology has led microelectronics industries to create a brand new class of machines, called Fuzzy Machines, to overcome the limitations of traditional computing systems when utilized as Fuzzy Systems.*
*This paper gives firstly an overview of the methods by which Fuzzy Logic data structures are represented in the machines (each with its own advantages and inefficiencies), then introduces WARP (Weight Associative Rule Processor) which is a dedicated VLSI megacell allowing the realization of a fuzzy controller suitable for a wide range of applications.*
*WARP represents an innovative approach to VLSI Fuzzy controllers utilizing different types of data structures for characterizing the membership functions during the various stages of the Fuzzy processing.*
*WARP dedicated architecture has been designed in order to achieve high performance exploiting the computational advantages offered by the different data representation adopted.*

## Section 1. Fuzzy Machines

Computer evolution is tending towards specialized machines which are optimized to meet the needs of particular languages or classes of problems. One result of this trend is that many systems now contain one or more general purpose processors supported by a variety of specialized devices (e.g. mathematical or graphical coprocessors) optimized for specific operations.
While the numerical computation field is comprehensively served by machines and specialized integrated components able to calculate numerical algorithms at very high speed and with great accuracy, there is little cost-effective hardware to support newer approaches to logic, particularly those involving non exact information processing.
In particular, the type of processing required to solve problems using Fuzzy Logic with its peculiar data structures cannot be effectively carried out on machines designed for completely different kinds of algorithms and data representations. To deal with the calculus involving the data structures of Fuzzy Logic such as Fuzzy Sets (with their related membership functions) and Term sets [1], Fuzzy Machines have been introduced.
With respect to the **functionality** these devices can be gathered into two main groups:

- **FUZZY COPROCESSORS**
- **FUZZY CONTROLLERS**

**Fuzzy Coprocessors** represent the equivalent of a general purpose machine with respect to Fuzzy calculus: they are the key for turning standard systems into Fuzzy Systems. These machines should not to be considered as the main processors of a system, but rather as an indispensable support in speeding up fuzzy applications.

**Fuzzy controllers** represent the next step in the evolution of intelligent controls and their use can lead to a technological breakthrough in this area. A Fuzzy controller is a particular Fuzzy device equipped with an interface suitable for driving physical actuators: it accepts deterministic values and produces a deterministic value.

With respect to the **technology** utilized a Fuzzy Machine can be implemented in the following ways:

- **SOFTWARE IMPLEMENTATION**
- **DEDICATED HARDWARE IMPLEMENTATION**
  - **HYBRID MACHINES**
  - **FULLY DIGITAL MACHINES**

The **Software implementation** of Fuzzy machines is presently the most widely used one; while this approach well suites off-the-line processing it becomes inadequate whenever processes requiring high or medium high dynamics appear.

Among the **Dedicated hardware implementations**, the **Fully Digital** approach to Fuzzy Logic Dedicated Machines is up to now the most widely employed method of implementation of dedicated machines [2], [3]. The advantages of this technology are the generally known ones:

- Complex data management architectures
- Easy interfacing in existing systems
- Low sensitivity to technology changes

The **Hybrid** (mixed Analog/Digital computing) realization of fuzzy machines may possibly represent the next edge in the computer world [4], in fact Hybrid machines provide a number of significant advantages over digital ones:

- Very high speed system throughput
- High parallelism allowed
- No need for expensive A/D and D/A converters

With this type of technology the problems mainly lie in the representation for the Fuzzy data structures, on the analog memories required by the machine and in the sizing of some components, but great improvements in those areas are expected in the next few years.

A rough picture of performances in terms of FIPS (Fuzzy Inference Per Second) obtainable with various types of platforms (and the type of applications were they are mostly applied) are illustrated in fig. 1.

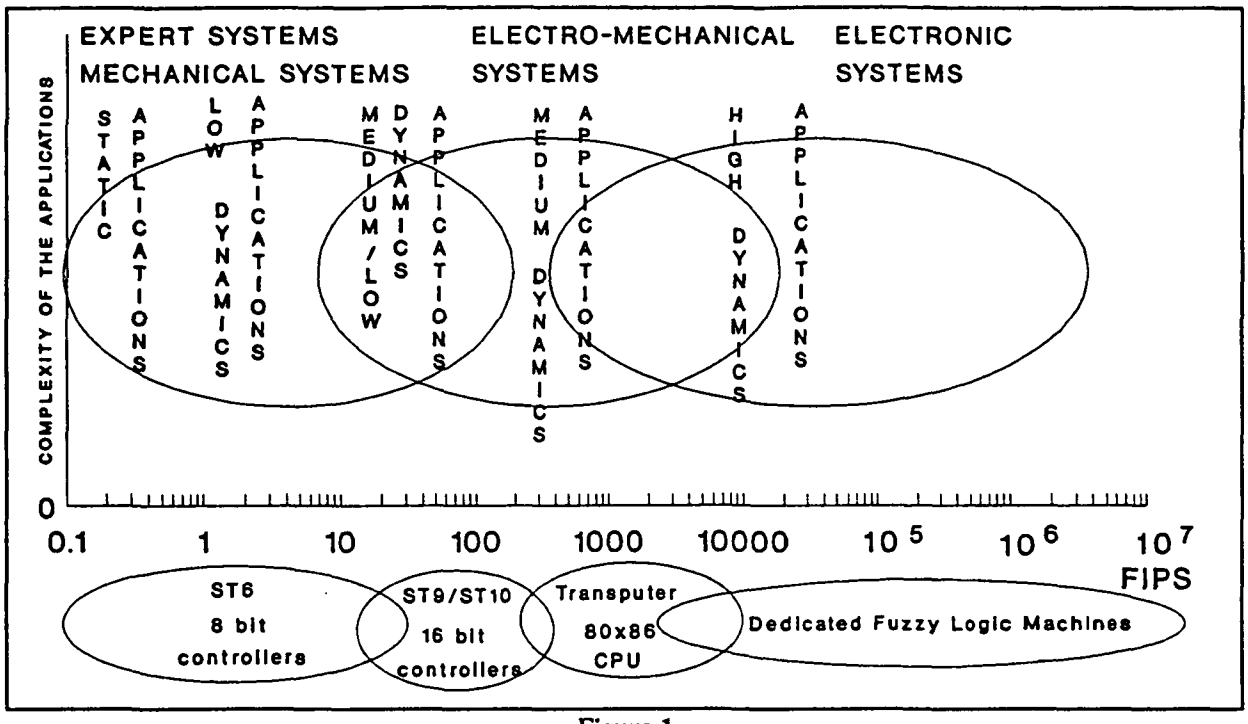


Figure 1

## Section 2. Design of Fuzzy Machines

Among the design approaches to fuzzy machines, and in particular the **Software** and **Fully digital** ones, a great advantage lies in the possibility of deciding during the **architectural design phase** the **"kind of machine"** that one wants to realize, ranging from the two opposite poles:

- **Memory oriented machines**
- **Computing oriented machines**

**Memory oriented machines** are characterized by having most of the computing performed off-line and then stored in suitable formats inside the memory. This lead to the utilization of large amounts of memory because the membership functions must be described by means of non-optimized data structures (in most cases vectors). Generally with this solution higher performances are possible although with a certain loss in precision.

**Computing oriented machines** come at the other end of the spectrum. Here the membership functions are stored in compact formats and it is the machine that must operate on those complex data structures performing all of the necessary computing (that is generally finding intersection points and calculating area/weight values).
This solution it is generally slower than the previous one but allows a greater precision.

The performances obtainable by the above approaches are greatly influenced by the level of internal parallelism that is actually implemented. It is worth noting that this parameter affects Computing oriented machines more than Memory oriented machines.

Another very important factor in the designing of the fuzzy machines, is the way of representing the membership functions; different methods can be utilized according to where in the rule (IF-part or THEN-part) the connected variable acts.

For the **membership functions** bounded to the **IF-part** of the rules there are two main types of representation that are commonly utilized:

- **Vectorial representation**
- **Analytical representation**

With the **vectorial representation** of the membership functions the universe of discourse is divided into a number of elements N, and the interval [0, 1] in L truth levels, creating a vector $\mu_1(x),...,\mu_N(x)$, where $\mu_i(x)$ represents the truth level that best approaches the value of the membership function $\mu(x)$ in the point i. With this type of characterization, the more critical decision is choosing the most appropriate values of N and L. Generally the values for N range from 25 to 256 and for L from 10 to 256, according to the type of technology utilized.

With the **analytical representation** a function that maps the universe of discourse onto the closed space [0, 1] is provided. This is generally a piecewise linear function described by the breakpoints where the function changes gradient. With this kind of characterization it is left to the machine to calculate the intersection point between the membership function and the function representing the input.

Clearly, the first method, characteristic of memory oriented machines, allows greater performance to be obtained as it is based on look-up tables rather than calculations. On the other hand the value of the intersection must be restricted to those realistically representable with the adopted technology, while in the analytical formalization values as precise as the machine data representation can be obtained.
The choice among the two above methods is generally a trade off between speed and precision.
The value computed from the IF-part of a rule is used to perform the inferential process on the membership functions of the THEN-part. To perform this operation a suitable inferencing method must be used. The two most widely employed ones are:

- **Max-Min inferencing method**
- **Max-Dot inferencing method**

The main difference between the two methods lies in the different truncation that is applied to the Membership Functions of the THEN-part of the rules. The choice between one of the above methods of inferencing is influenced by the type of representation of the Fuzzy Sets adopted. The Max-Min Inference method truncates the membership function up to the threshold value ⊛ while the Max-Dot Inference utilize the value as a scaling factor. This is clearly explained by fig 2.

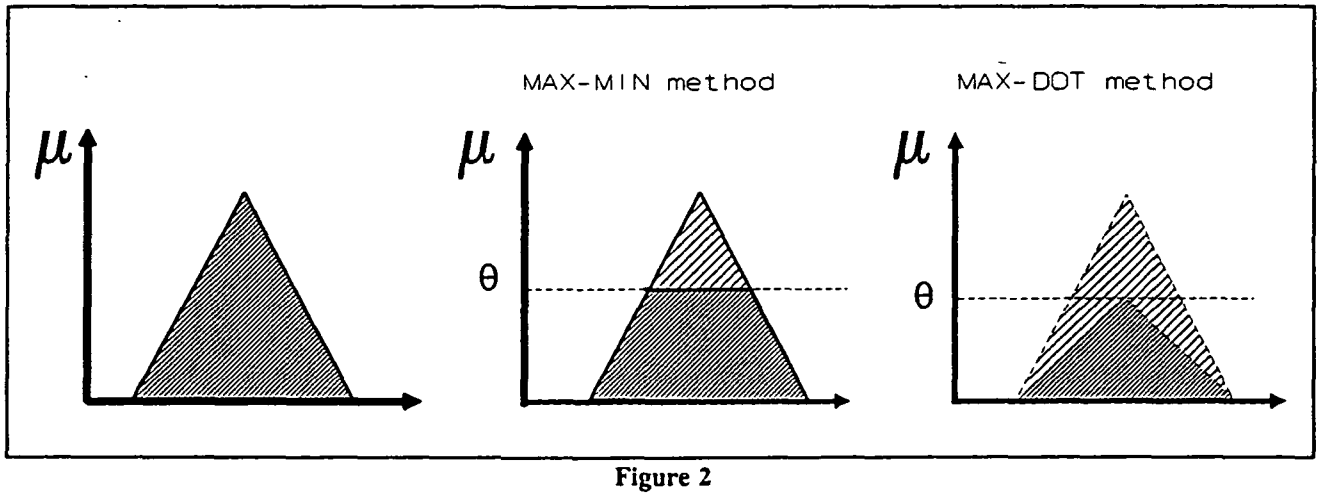


**Figure 2**

The Max-Min Inference is mainly adopted when the membership function is defined through a vectorial representation, in fact in this case it is relatively easy to compare each value of the M.F. with the threshold value and choose the smaller. The Max-Dot method it is not so easily performed because it is necessary to multiply by the scaling factor each non-zero component of the vector. Conversely, the Max-Dot inference method is preferred with the analytical representation, as it is easy to calculate the resulting M.F. by multiplying each breakpoint value by the scaling factor, whereas the Max-Min method requires a new series of breakpoints to be calculated.

There is a third method of representing the membership functions of the THEN-part in the particular case of Fuzzy Controllers, where the output of a Fuzzy inference is not used as input for another. In this case, the M.F. representation can be reduced to the only two parameters that are effectively needed in the assembling and defuzzification phase: a weight representing the area underlined by the M.F. and its point of application (barycentre). In fact the defuzzified output comes from a linear combination of those values, as clearly illustrated in the defuzzification algorithm generally adopted:

$$C^* = \frac{\sum_{i=1}^{MF_n} A_i' \cdot Xg_i'}{\sum_{i=1}^{MF_n} A_i'} \qquad (1)$$

$Xg_i'$= Barycentre of the $i$ M.F. truncated at the $\Theta$ truth level.

$A_i'$= Area of the $i$ M.F. truncated at the $\Theta$ truth level.

$MF_n$= Number of M.F.s of the output

The inferencing method chosen strongly influences the way in which the M.F. are assembled prior to the defuzzification phase. Essentially the two methods commonly adopted differ in the treatment of the zones of the universe of discourse where two or more M.F.s have non-zero values.
Fig. 3 shows the two approaches: in 3(A) the resultant M.F. is obtained by taking the greater of the two component values at any point whereas, in 3(B) the combined M.F. is obtained by simple addition of the component values. In effect, 3(A) represents a logical combination and 3(B) an arithmetical combination of the two M.F.s.

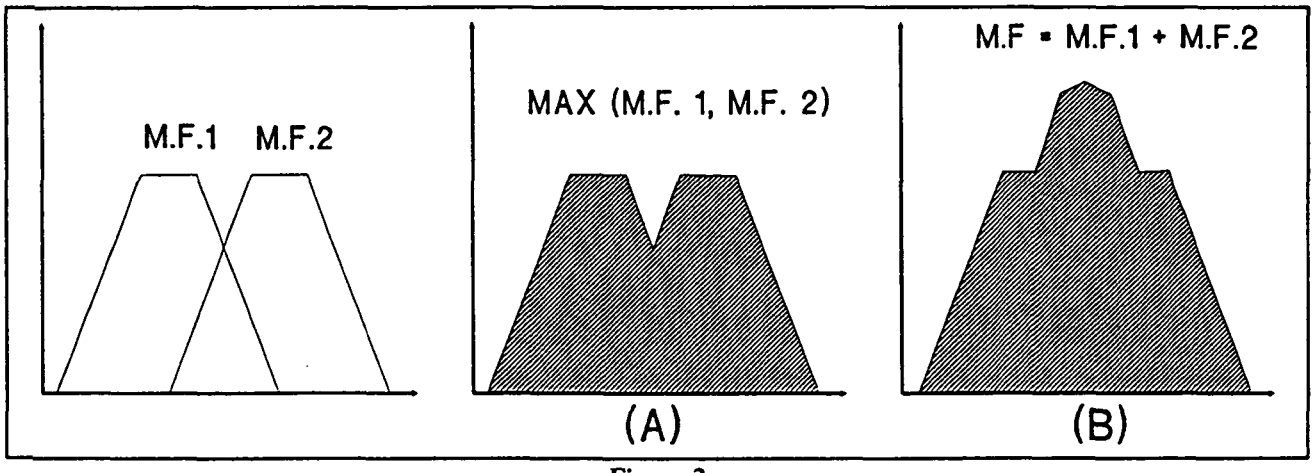


**Figure 3**

Depending on the method of representing the M.F.s it is possible to choose between the two above methods of assembling: only the arithmetical combination is allowed with the weight/barycentre representation while either of the two assembling methods can be chosen with the other two representations. however, the way in which M.F.s are to be represented and combined greatly affects the machine architecture, so these decisions must be made at an early stage in the design of a particular Fuzzy Machine.
It appears clear from what has been presented above, that an efficient general purpose fuzzy machine cannot exist but rather one must rely on machines tailored to meet the needs of a particular class of problems.

## Section 3. WARP: Weight Associative Rule Processor

WARP is a dedicated VLSI machine whose architecture has been designed in order to efficiently exploit all the advantages of Fuzzy calculus. The major innovation with respect of traditional approaches to Fuzzy Machines has been the adoption of **different data structures** for the various phases of the computational cycle. In fact one of the greatest limiting factor in the traditional fuzzy architectures is the use of the same data representation for both the computation connected to the IF-part and to the THEN-part of a rule.
In order to represent the membership functions connected to the Fuzzy variables of the antecedent of the rules we adopted a **vectorial representation** of the Membership Functions based on 64 ($2^6$) or 128 ($2^7$) elements, each possessing 16 ($2^4$) truth levels.

The utilization of vectors for this phase of the Fuzzy calculus has the great advantage that in the case of a controller, for each rule the data involved in the computing are one or more M.F's (representing the knowledge of the system) and one or more crisp values (representing the input from the "external" world).
With this data representation, in order to find the matching level (hereafter called $\alpha_i$) between the input and the stored M.F.s it is sufficient to get the various $\alpha$ corresponding to the truth level of the element located by the projection of the input in the universe of discourse. Classically the vectors characterizing the membership functions of a term set are stored sequentially in the memory as illustrated in fig. 4.

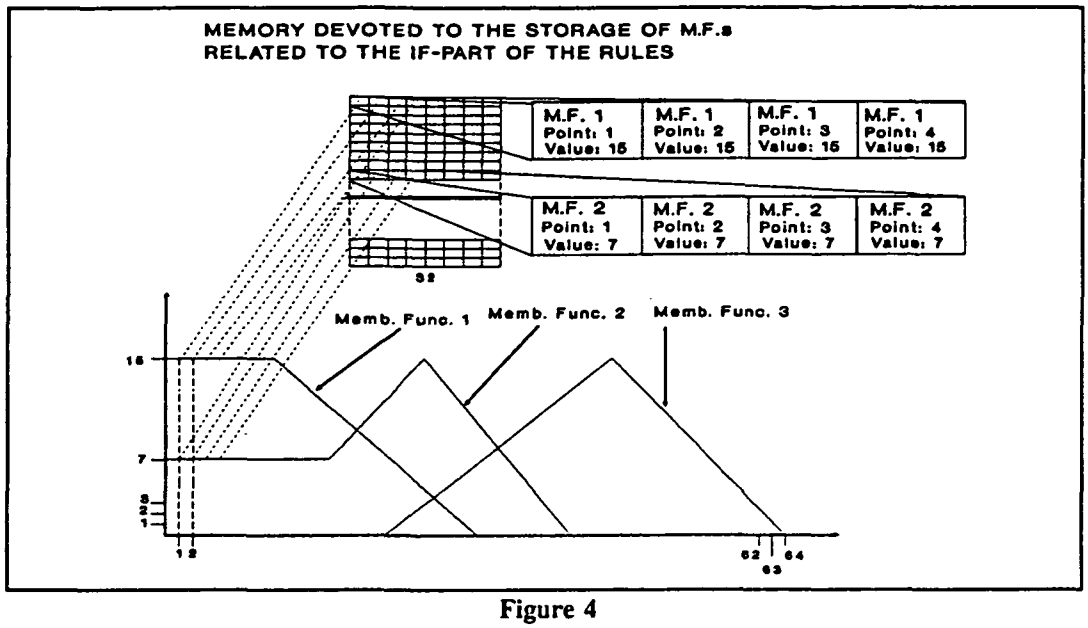


Figure 4

In this situation it is necessary to independently address each memory word containing a needed $\alpha$ value. The number of memory accesses is thus a function of the membership functions comprising the term set. The memory access time being one of the most critical parameter of the computation, it appears clear that in order to obtain high performance the number of memory accesses must be reduced as much as possible.
In order to efficiently perform the computation of the IF-part of the rule, WARP architecture has been built up around a different idea for storing the membership functions. The WARP approach consists in storing in successive memory location of the same memory word all the $\alpha$ values comprising a term set. This term set is formed by the membership functions connected to the IF-part of the rule, as showed by fig. 5. In this way it is possible to retrieve all the $\alpha$ value of a term set using the crisp input value to calculate the memory word address in the fuzzy memory device utilized.

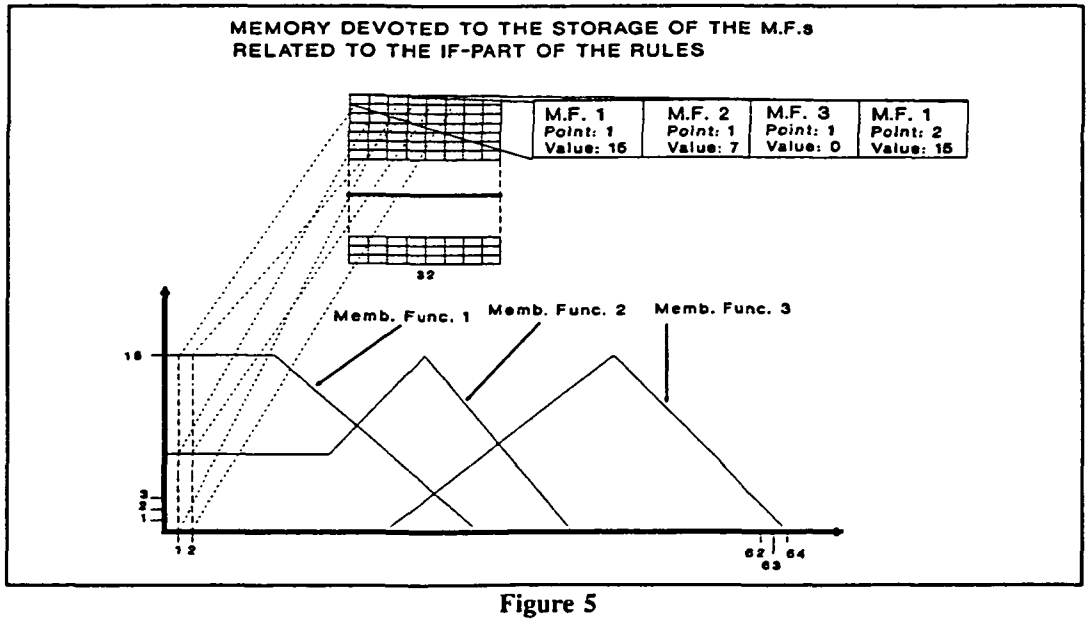


Figure 5

The number of memory accesses is a function of the M.F.s comprising a term set and inversely proportional to the size of the memory word, obtaining a significant reduction of the number of access in comparison with the traditional information storage methods. Assuming memory words with the same width (32) and elements of the vectors with the same characterization (4) the number of accesses is reduced by a factor of 8 (32/4).

Although the illustrated method for storing and retrieving the various $\alpha$ values connected to a fuzzy variable is highly efficient, once the related $\Theta$ value (the truth level for modifying a variable of the THEN-part) has been calculated, the vectorial computation becomes slow due to the huge time consuming process of modifying the M.F.s of the right side of the rule with the threshold value provided and assembling all the M.F.s that will form the M.F. furnished as output. Taking in account some limiting factors like:

- The number of parallel computational elements that realistically can comprise such a device
- The linear increase in memory size when trying to augment the number of elements which characterize a M.F
- The necessity to cycle over all the elements of the M.F. provided as output in order to carry out the defuzzification phase

It is clearly apparent how inefficient is such information management.
WARP avoids the above limitations. Having a limited number of possible truth values (15 excluding 0) coming from the IF-part of a rule, it is possible to represent a membership function connected to the THEN-part utilizing 15 words of memory, each containing both the value (weight) of the area underlined by the M.F. and the point of application (barycentre). In order to achieve a more efficient computation, for each memory word characterizing a truth level WARP directly stores both the area multiplied with the barycentre and the area itself, as illustrated in fig. 6.

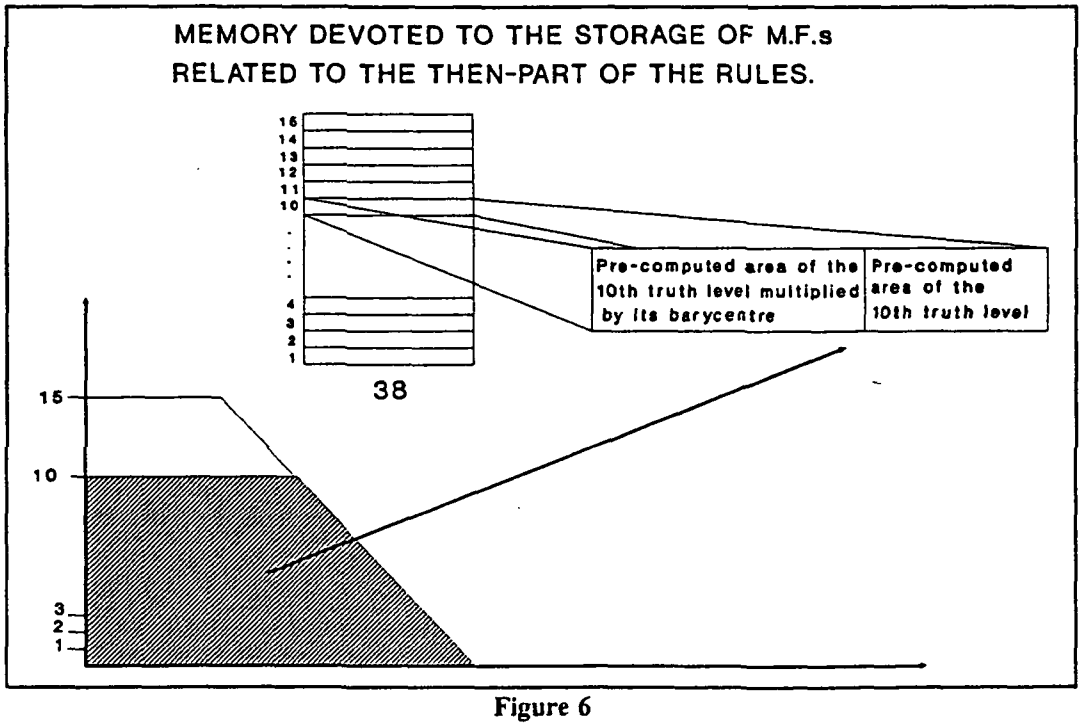


Figure 6

With such a method for storing information, the inferencing method adopted (Max-Min or Max-Dot) is perfectly transparent with respect of the computational architecture, in fact the only difference between those methods lies in the different value of the area of the resulting M.F. as clearly illustrated in fig. 2.
A great computational advantage of the approach is that a great part of the fuzzy computing can in effect be performed off line. The particular data structure adopted in WARP for representing the M.F.s allows an assembling methods of type (B) with reference to fig. 3.

Section 4 **WARP architecture.**

**WARP** is a **VLSI Megacell** whose architecture has been designed in order to be employed in different environments. The dedicated memories and the computing blocks have been defined with the purpose of efficiently operating with a representation of the membership functions as previously illustrated.
The architectural data flow/block diagram of the **Fuzzy megacell core** is shown in Fig. 7.

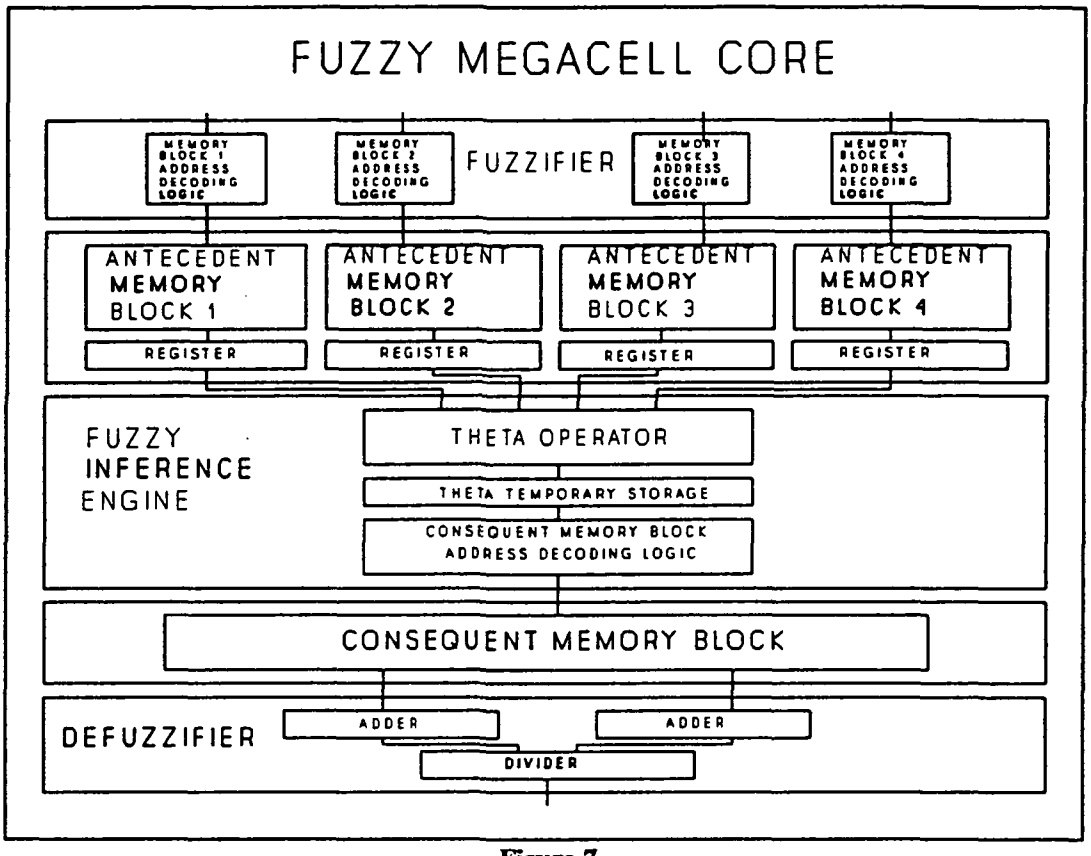


**Figure 7**

The **Fuzzifier section** is devoted to the calculus of the memory address corresponding to an input and the retrieving of the stored information. The assumption of always expecting as input a crisp value combined with the particular storage method has allowed the fuzzifier to be reduced to its simplest structure.
To obtain high performances the **memory** devoted to the storing of the membership functions of the **IF-part** of the rules has been divided in 4 independent blocks. Each of these blocks contains all the $\alpha$ values of one or more fuzzy variables, allowing the parallel retrieval of the $\alpha$ values. Inside the memory block, the data representing the membership functions are stored according to the scheme of section 3.
This splitting of the memory has also induced the necessity of also having 4 fuzzifier sections (one for each memory block). The $\alpha$ values found are memorized in a set of devoted register and then opportunely processed to calculate the $\Theta$ value of each rule.
The adoption of the vectorial data representation for the M.F.s of the IF-part of the rules allows this operation to be performed in an highly efficient and flexible way inside the Fuzzy Inference Engine via the **Theta-operator**, whose block diagram is illustrated in fig. 8. This operator has been designed in order to carry out operations with an unlimited number of terms connected by OR and/or AND connectives. This block is utilized mainly to augment the performances of the device, in fact practically all the Fuzzy computing is performed here (the defuzzification although computationally heavy cannot be properly classified as fuzzy computing).

The $\Theta$ values are used to calculate the **address of the memory word** in the **memory block** where the membership functions bounded to fuzzy variables of the **THEN-part** of the rules are stored. Inside this memory block, the values of the M.F.s are stored with the technique illustrated in fig. 6.
The memory block devoted to the fuzzy variables of the **THEN-part** of the rules has not been divided because the computational requirements and the architectural simulations have clearly shown that the addition of dedicated hardware is not balanced by a significant increase in performance.

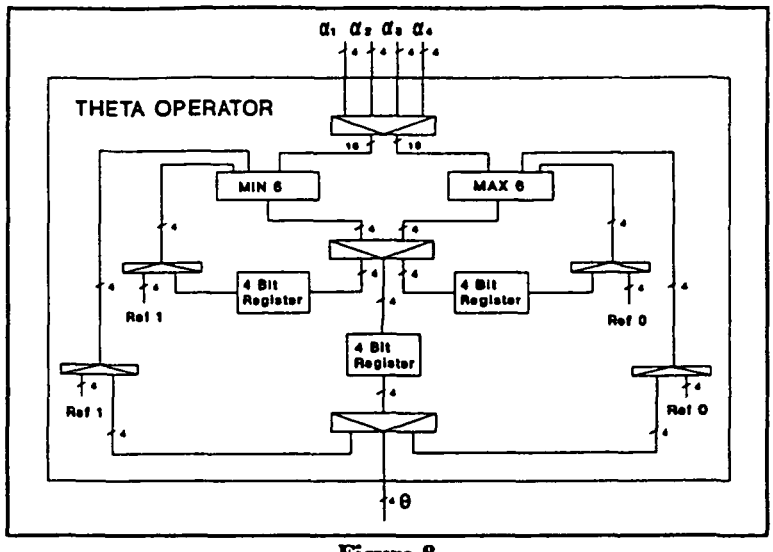


**Figure 8**

The assembling of all the membership functions comprising an output and the defuzzification process are carried out in the **Defuzzifier block**. Thanks to the particular representation of the membership functions, this phase can be performed with a limited number of operations. The studied architecture utilizes 15 memory words, each 38 bit wide, to store the relevant information of each M.F. Having adopted the defuzzification algorithm previously illustrated in section 2, a saving of 2 or 3 multiplying operations is obtained (actually those necessary to calculate $A_i'$ and $A_i' \cdot Xg_i'$) with related hardware and, most of all, a great freedom in defining the M.F.s themselves is allowed. In fact in this way a membership function doesn't need to be symmetrical as would be the case if it was described giving only the whole weight and its barycentre. With the adopted method each truth level is characterized by the actual weight and its point of application thus effectively overcoming any constraint related to symmetry.
The Fuzzy megacell can be employed in different environments. The ST9 microcontroller thanks to its flexible architecture is well suited to being augmented as in the configuration illustrated in fig. 9.
In this way the microcontroller can perform normal control task while WARP will be responsible for all the fuzzy related computing in independent mode.

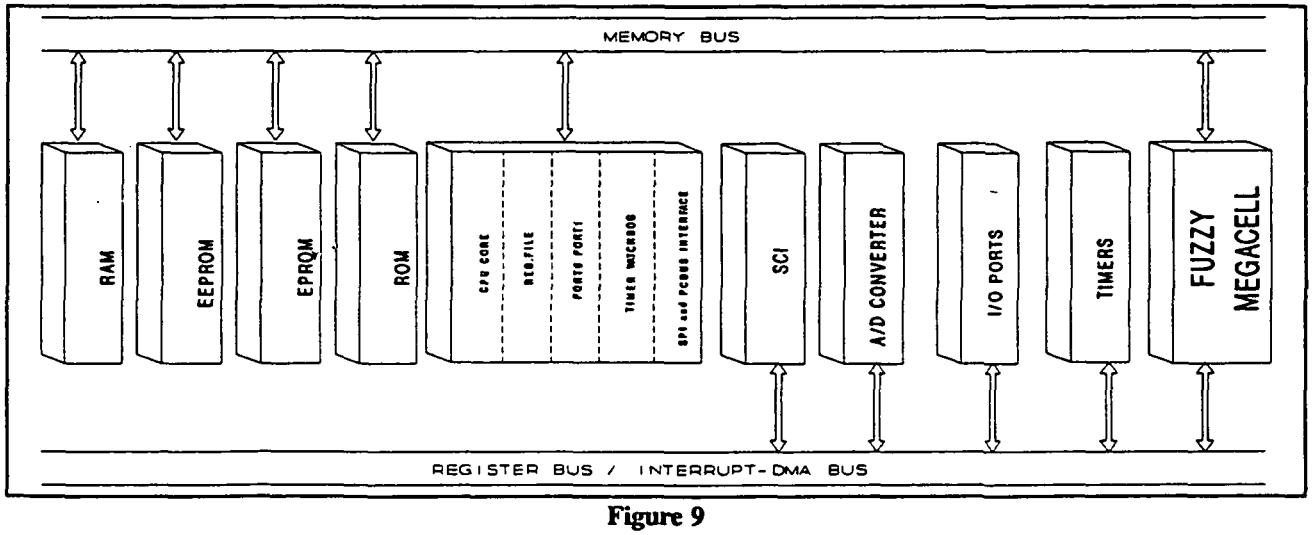


**Figure 9**

The Fuzzy megacell can also be configured as an embedded controller in a configuration as the one illustrated in fig. 10.

WARP is currently in the advanced design phase. In order to guarantee high compatibility with customers needs and assure maximum flexibility, a TOP-DOWN design methodology has been adopted for it and the VHDL language to implement it. VHDL (VHSIC, or Very High Speed IC, Hardware Description Language) is the IEEE standard language for the description and simulation of electronics circuits. WARP hardware structures have been synthesized utilizing SGS-THOMSON's own 0.8 μm technology. The subsequent structural simulations have displayed performances in the order of 10 MFIPS.

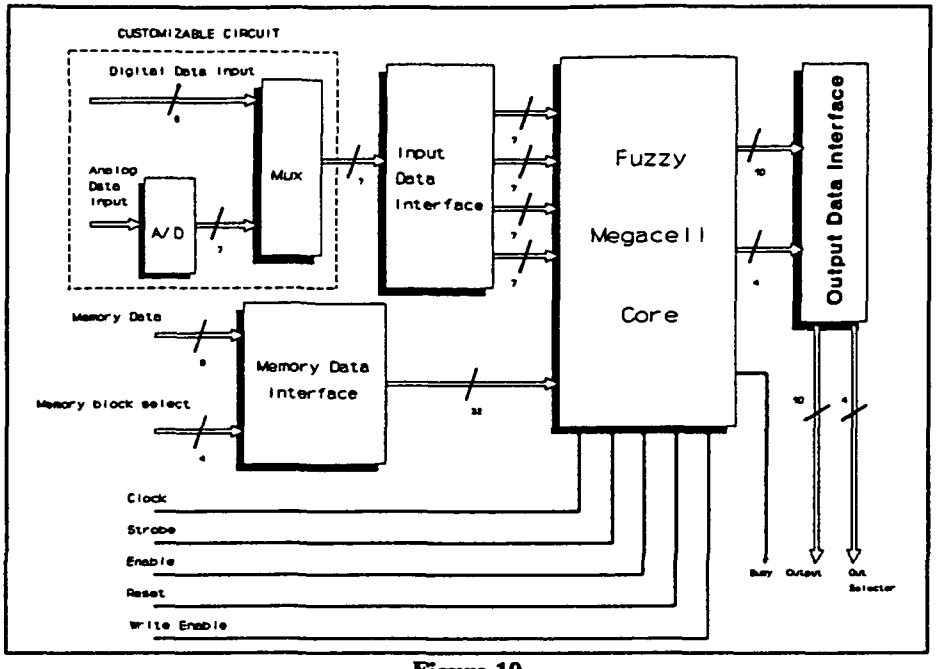


**Figure 10**

## Section 5 Conclusions

In order to provide an answer to a wide number of application requests, WARP design relies on concepts of flexibility and modularity. The innovative approach of WARP is represented by the adoption of different data structures to represent the membership functions characterizing the fuzzy variables of the left and right sides of the rules. Great emphasis has been put on granting the user maximum flexibility in defining the membership functions. This has been carried out allowing the definition of Term Sets with no fixed numbers of fuzzy sets; moreover the possibility of defining the single membership functions without any constraint like symmetry/shape proved very useful in characterizing complex control applications. The careful analysis of the computational requirements during the various stage of the fuzzy processing and the subsequent mapping in adequate hardware structures has lead to the achievement of high level of computational efficiency permitting performance in the order of 10 MFIPS to be obtained while reducing the number of parallel computational elements. Moreover the architecture is totally transparent with respect of the types of memory utilized (EEPROM, Flash ...) and technology (Sub-μ CMOS, ...) so allowing the device to be used for a wide range of applications.

## Section 6. References

[1] L. A. Zadeh, "Fuzzy Logic", Computer, April 1988

[2] H. Watanabe, W. D. Dettloff and K. E. Yount, "A VLSI Fuzzy Logic controller with Reconfigurable, Cascadable Architecture", IEEE Journal of solid-state circuits. vol 25, No. 2, April 1990.

[3] H. Ikeda, N. Kisu, Y. Hiramoto and S. Nakamura, "A Fuzzy Inference Coprocessor Using a Flexible Active-Rule-Driven Architecture", pp. 537-544, IEEE International Conference on Fuzzy Systems 1992

[4] T. Yamakawa, "Intrinsic Fuzzy electronic circuits for sixth generation computer" in Fuzzy Computing, M.M Gupta and T. Yamakawa, pp. 157-171, Elsevier Publisher B.V. (North Holland), 1988

[5] A. Pagni, R. Poluzzi and G.G. Rizzotto, "WARP: Weight Associative Rule Processor. An innovative Fuzzy Logic Controller", pp. 543-546, Proceedings of IIZUKA'92 2nd International Conference on Fuzzy Logic and Neural Networks, 1992