A two input fuzzy chip running at a processing rate of 30 nanosecond realized in 0.35 micron CMOS technology

DAVIDE FALCHIERI, ALESSANDRO GABRIELLI, ENZO GANDOLFI, MASSIMO MASETTI

Physics Department
University of Bologna and Istituto Nazionale di Fisica Nucleare
viale Berti Pichat 6/II 40127 Bologna
ITALY

CSCC'99 Proc.pp.2751-2758

Abstract: - In these years we have been dealing with the problem of VLSI fuzzy chip design because the processing rate of the fuzzy chips available on the market is too low for trigger applications in High Energy Physics Experiments. So far we have designed a two input fuzzy processor in VHDL code using the Synopsys SW as front end tool for synthesis and optimisation of the VHDL code. The chip has been targeted to the Alcatel Mietec 0.35 micron CMOS technology since it is one of the most powerful and it is available at an accessible price through Europractice. The chip architecture is pipelined with a clock frequency of 133 MHz that means a processing rate of 30 nanoseconds since only the active rules are processed. The chip has been sent to IMEC the last December to be constructed. The chip size has an area of 3 mm^2 for a total power consumption of 200 mW: the place and route has been done at IMEC with the Avant! software. Proc.pp.2751-2758

Key-Words: VLSI, CMOS, Fuzzy, Fuzzification, Defuzzification, Active Rule Selector, Membership Function

1. Introduction

There are applications in High Energy Physics Experiments where it is necessary to reach a high processing rate such as few tens of nanoseconds. Since we decided to face the trigger problem using a fuzzy-logic based system we decided to design a dedicated chip. In fact most of the fuzzy logic based controllers available on the market [1] [2] are much more slower than the ones required in trigger applications. We decided to have our chip realised using Alcatel Mietec 0.35 μm CMOS VLSI technology, one of the most interesting among the deep sub-micron ones available through Europractice. In order to reach similar performances we have taken the following decisions:

- to design a parallel-pipeline architecture of the chip at the highest possible frequency (the technology adopted allowed us to reach a maximum frequency of 133 MHz);
- to implement in HW the simplest algorithms like the Sugeno order zero inference and defuzzification method [3];
- to process only the active rules, when possible [4] [5];
- when not possible we decided, for our applications, to design fuzzy processors matched to the fuzzy systems obtained by a genetic rule generator [6].

We want the processing rate to be independent from the fuzzy system that has to be processed. To get this result:

- the fuzzy system is previously converted into an equivalent one where all the rules are present (each one involving all the variables) and then loaded into the on-chip rule memory;
- no more than 2 adjacent MFs can overlap;
- the rule antecedent allows to determine the rule address;
- an Active Rule Selector, ARS, is able to identify and process only the active rules related to each input data set without time consuming [7].

2. The chip main performances

The chip architecture involves 12 pipeline stages, each one taking 7.5 ns.
The main features of the fuzzy processor can be summarised as follows:

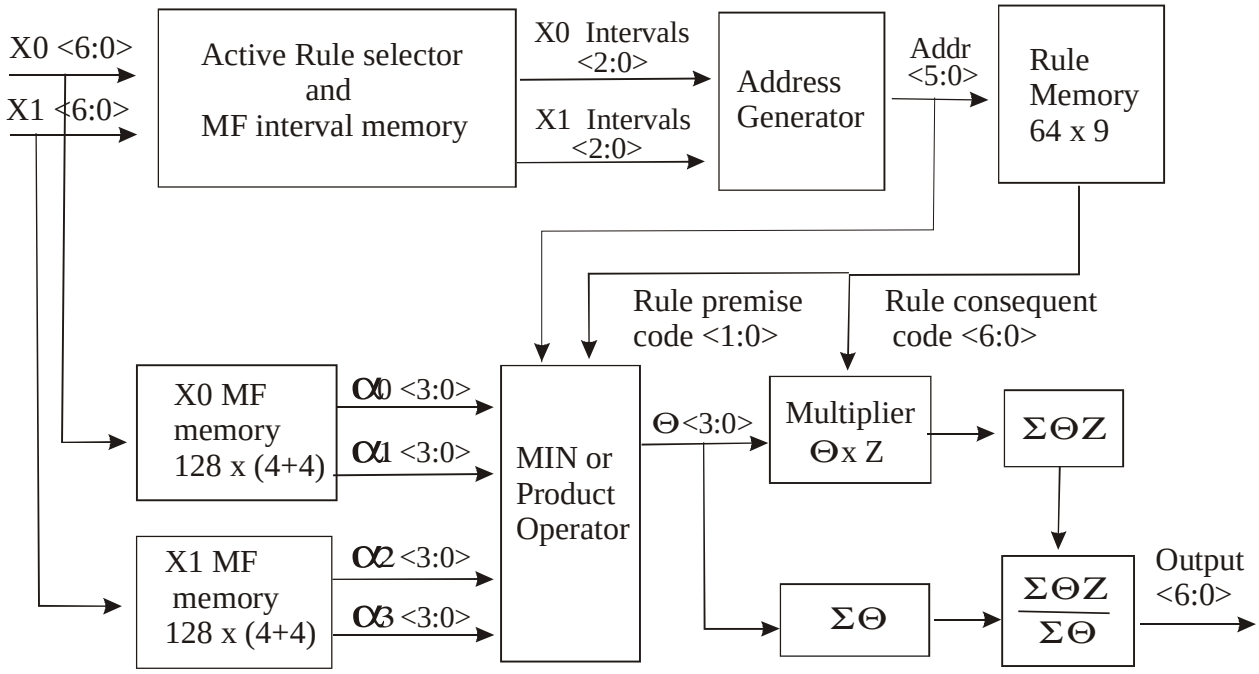

Figure 1. The fuzzy chip architecture

- two 7 bit inputs, one 7 bit output;
- 8 membership functions for each of the 2 input variables;
- maximum overlapping of the input MFs no more than 2;
- 128 crisp MFs for the output variable;
- 4 bits both for the antecedent and the premise degree of truth, from now on called respectively α and θ value;
- T-norm implemented by Minimum, (MIN), or Product to obtain the θ value;
- Sugeno order zero inference and defuzzification method;
- 133 Mega Fuzzy Inferences per second with a 133 MHz clock - this is the maximum clock speed allowed by the RAM memories in the Alcatel Mietec 0.35 μm technology.

With these features we obtain the following performances:

- any mathematical function can be approximated with an error of 1%;
- each rule is processed in one clock period;
- the processing rate is: number of active rules times the clock period: 4 x 7.5 ns = 30 ns;
- the total processing time is: processing rate plus the following contributions:
  - number of periods for the input synchronisation: 1 x 7.5 ns = 7.5 ns;
  - number of the pipeline stages times the clock period: 12 x 7.5 ns = 90 ns;
  - number of periods required by the division process: 4x 7.5 ns = 30 ns.

The main chip architecture is reported in figure 1. The Active Rule Selector selects the active rules related to the actual values of the input variables.

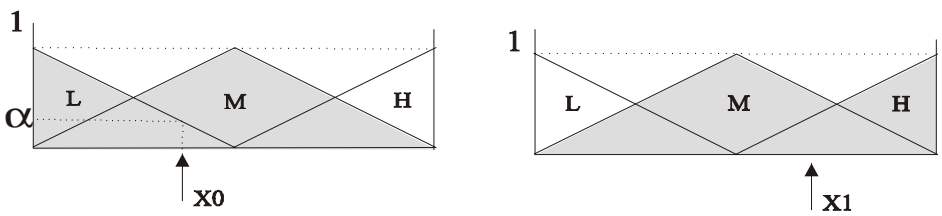

Figure 2. An input data set and the related MFs

To do that it generates the related addresses for

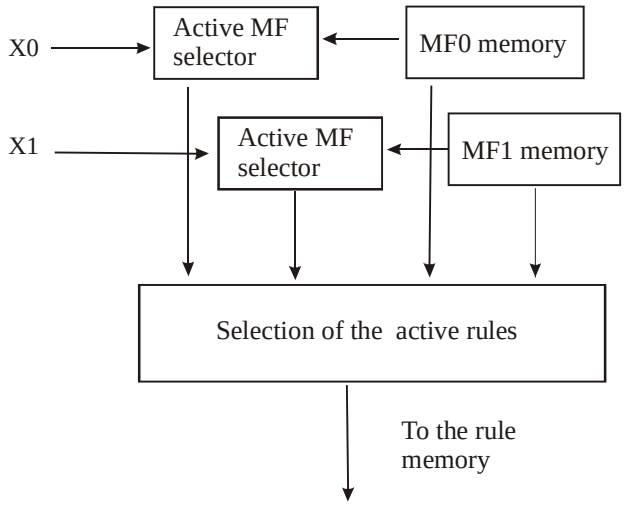

Figure 3. The active rule selector

the rule memory. Then the fuzzification process starts. Here follows an explanation regarding the way the rules are stored and how the active ones are selected, then each block will be described in more details.

3. The selection of the active fuzzy rules

To understand what an active rule is let us suppose to have a fuzzy system with 2 input variables, 3 FSs for each input with an overlapping of the MFs not higher than 2. A typical fuzzy rule for the above fuzzy system is like:

if (X0 is Low) and (X1 is Medium)
then (Z is High)

You can see in figure 2 that only the four rules where X0 is related to L or M FSs and/or X1 to M or H FSs give a non zero contribution to the final result, that is the α values are different from 0: these rules are called *active rules*. Figure 2 shows the α value which is the degree of truth related to the predicate X0 is Low. If we have N input variables, K FSs for each input variable, only t-norm operator for the rules and at most an overlap of 2, the active rules are 2^N while all the possible fuzzy rules are K^N. Therefore, for the above example, we can process only 4 rules instead of 9 for a fuzzy system made of all the rules.

Our 2 input fuzzy processor is able to process only the active rules, which are 4, between all the possible ones, which are 64 without time consuming; to obtain this result our solution requires:

- to store a fuzzy system made of all the possible rules in the fuzzy chip;
- to use the premise code of a rule as address of the same one, where a specific code is stored as described below.

To illustrate our rule code let us suppose to have only three MFs for each of the two input variables. In that case the number of all possible rules is 8. Then we store the rules starting from the address 000 where it is stored the rule:

if (X0 is Low) and (X1 is Low) then (Z is ...)

and at the address 001 the rule:

if (X0 is Low) and (X1 is Medium) then (Z is ...)

and so on.

The rule code consists of two parts: the first one, related to the premise, where 0 means that the related variable is not present in the rule while 1 means the vice versa and the second one, related to the consequent, reports the crisp output fuzzy set value. In this way the rule address defines the fuzzy sets of the premise while the rule code confirms or not if the rule was present in the initial fuzzy system (the one that might not contain all the rules) and which input variables and output FS were involved. For example the premise code "10" means that the active rule involved comes from a rule in the former fuzzy system where X0 is present while X1 is not and the consequent code identifies the crisp value of the output FS. If we had "00" as a premise code it would mean that there is not a corresponding rule in the initial fuzzy system, therefore its contribution must be zero.

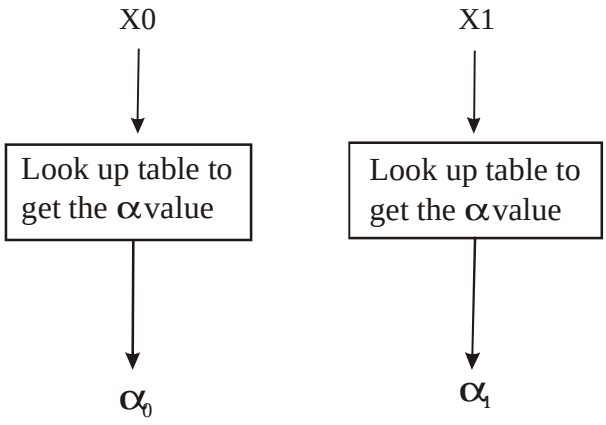

Figure 4. The fuzzification block

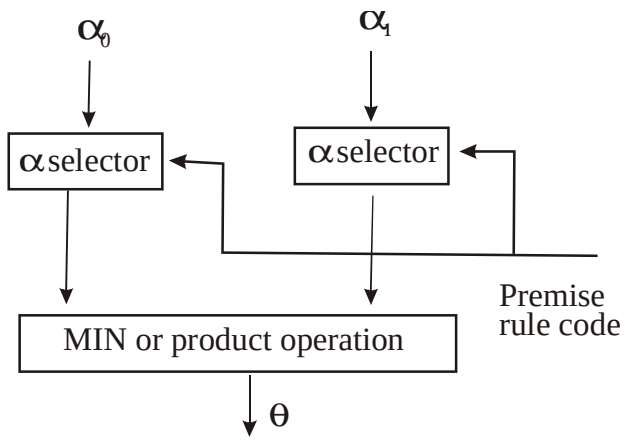

Figure 5. The MIN or Product operator

4. The chip architecture

Firstly the input data set has to be loaded into the chip according to the input synchronisation handshake signals. The input data set must be ready at any time depending on the external device that generates it but it has to be loaded just when the fuzzy processor is ready itself and has also to be synchronised with the fuzzy chip clock signal. For this reason the fuzzy processor has been provided with an Input_Ready and an Output_Ready handshake signals. The chip architecture is made of several blocks that will be here illustrated.

4.1 The Active Rule Selector

In this block there are two RAM memories, see MF0, MF1 modules in figure 3, which contain the beginning and ending points of the 8 MFs related to each variable. For each input data set it proves sufficient to store only six words because for 8 MFs there are 7 intervals and the beginning and ending point are known. Then an active MF selector can select the two active MFs related to each input variable actual value. As soon as these MFs are identified another circuit is able to compute the address code of the related active rules.

4.2 The fuzzification process

The fuzzification process, that is the calculation of the input degrees of truth, may be accomplished in either of two different ways: or with an arithmetic calculation [8] or simply using a lookup-up table. As an arithmetic calculation can be quite time and area consuming, we decided to implement a fast look-up table method, see figure 4. A second advantage of this method is its high flexibility in defining membership functions: any arbitrary membership function shape can be defined. Besides the restrictions on the maximum overlapping of two fuzzy sets leads to a very small memory block.

4.3 The MIN or Product operator

Via an input pin the operation mode is selected, that is the θ value can be computed either with the minimum operation or with a product. The MIN or product circuit receives the α values and the premise code of the rule being processed, which is related to the fuzzy system where all the rules are present, as in figure 5. That rule may have or not a corresponding rule in the original fuzzy system: if it has the corresponding one it may have or not all the input variables. Therefore the premise code takes into account if that rule is present: for example if there is only X1 the premise code is "01". Therefore the α selector circuits generate for the α0 the 1111 number to be processed by the MIN or product operator and so on.

4.4 The defuzzification block

This block receives from the MIN circuit the θ value and from the rule memory the consequent code, that is the crisp 7 bit Z value and it

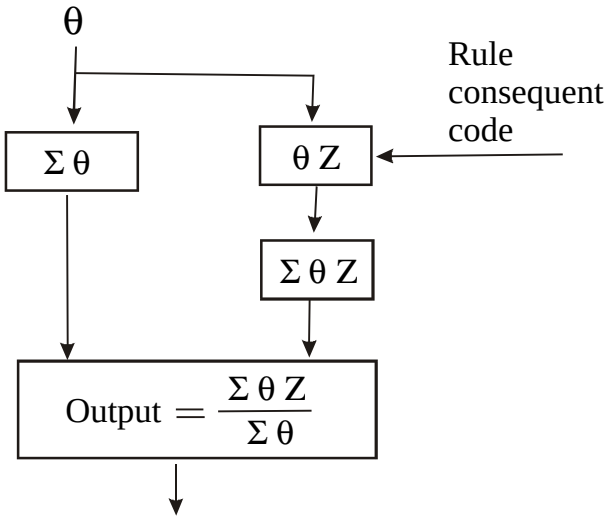

Figure 6. The defuzzification block

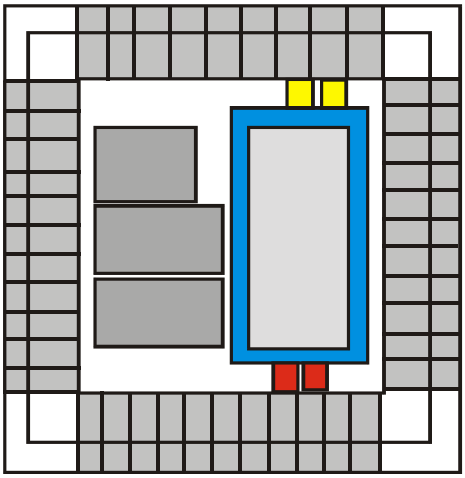

Figure 7: The block diagram of the chip layout

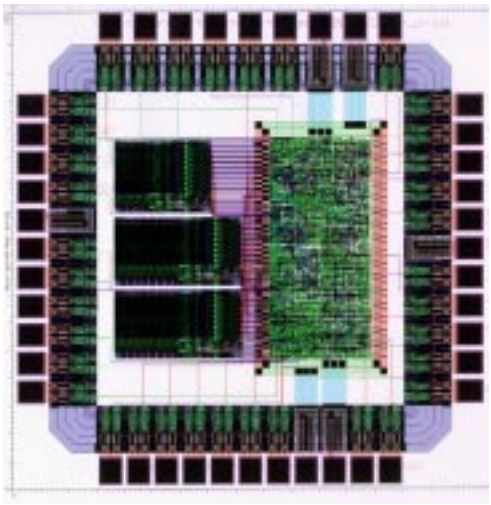

Figure 8. The fuzzy chip layout

computes the $\Sigma\theta$ and $\Sigma\theta*Z$ operations. After processing the last rule a new input data set can enter and the division process starts to compute, in parallel to the pipeline processing, the Z_o output value (see figure 6):

$$Z_o = \Sigma Z_i * \theta_i / \Sigma \theta_i \quad (1)$$

4.5 The pipeline timing

We will describe now what elaboration takes place, clock by clock, in each pipeline stage:
- at the first clock a new input data set is loaded into the input register; this process is off line to the pipeline processing;
- the ARS requires one clock cycle to generate the code of the active MFs, the MFs corresponding to the actual values of the inputs;
- the fuzzification block requires 8 clock periods to compute the α values: pipeline steps from 2 to 9;
- the MIN or Product block requires one clock to compute the θ value: pipeline step 10;
- the defuzzification block requires one clock to compute the $\Sigma \theta_i$ and the Sugeno multiplication Z_i θ_i operations and another clock to compute the $\Sigma Z_i * \theta_i$ operation: pipeline steps 11 and 12.
- the division process takes place when the last rule has been processed and runs in parallel to the pipeline stages. It takes nearly 30 ns.

5. Implemented Features for I/O Control

As already explained above, for HEPE applications the speed, in terms of computation time, is a very important constraint and is absolutely to be met; then for having a flexible fuzzy chip, it also has to be easy-to-use for what concerns the input-output handshake signals.
The fuzzy processor is to be used mounted on a printed board and synchronised with an on-board clock signal. So far it provides itself with all the synchronisation steps between itself and an external device write and load cycles.
The fuzzy processor, in fact, does not delegate the input-output handshake synchronisation signals to external devices such as controllers or dedicated processors, but a simple handshake signal configuration has been studied. Here follows a brief description of these input-output signals:
- an Input_Ready signal, synchronous with the on-board clock signal, is used for enabling the external device write cycle. In other words the external device can write its data into the fuzzy processor internal RAMs, by means of an external driven Load_Input signal, just when this Input_Ready signal is active. In addition, the external device must hold the input data set and the Load_Input signal valid for at least two on-board clock periods. In this way the fuzzy processor can both recognise the external device write cycle and synchronise the device data with the on-board clock signal;
- one output signal named Output_Ready has been implemented to enable the external device for accepting the output data of the fuzzy processor. Since the fuzzy processor may be synchronised with a up to 133 MHz (7.5 ns) clock rate, and since the division process can take up to 30 ns, these output handshake signal is synchronised

four clock periods after the division process has started.

6. The VLSI Implementation

From the point of view of the VLSI implementation of the fuzzy chip we have divided the chip in blocks which are different from the ones described in the previous paragraph: for example the rule memory is one of the most important because of its size. In figure 7 a block diagram of the chip layout is shown while figure 8 reports the actual layout. These blocks have been designed in order to reach both high clock speed and low power consumption. Of course a trade-off solution has been adopted but, anyway, the proposed goals have been met.

In 1997 we designed and constructed a first version of the two input fuzzy processor [9] in ES2 0.7 μm CMOS technology with a 50 MHz clock. The chip had a total area of 14 mm^2 and a total power consumption of 1 W. Moving from 50 to 133 MHz has required some architectural changes to the old structure: it is not sufficient to re-synthesise the old VHDL code on the new technology, as one could believe. First of all the pipeline stages have to be optimised for a clock period of 7.5 ns: this means that some of the blocks have to be subdivided into faster ones until they can work at the target speed. For example the multiplier θ * Z (4 x 7 bits) has been subdivided into two smaller blocks working in pipeline at the right frequency of 133 MHz. The total latency has increased to 2 clock periods but every 7.5 ns a valid data θ * Z is produced in output. So far the total number of pipeline stages will increase even if the final latency will not. Another useful guideline for designing ASICs over the frequency of 100 MHz is to strictly avoid the use of gated clocks. Clock gating proves to be useful when some blocks of a circuit have to be enabled while others not. For example we made use of this technique in the old version of the fuzzy chip for selecting only one of three memory blocks at a time (while the others two are in a stand-by state) in order to save power. While this technique is widely used for low-medium frequencies it may generate some risks at higher speeds. In fact using combinatorial control logic on the clock net may give rise to spikes whose effect may randomly affect the behaviour of the entire logic of the circuit. In this new version of the fuzzy chip the clock net directly drives all the 300 flip-flops and the 3 RAM memories without any control logic on it. One has only to provide a clock buffer able to drive this net with an acceptable skew: this means that the clock signal has to reach every flip-flop in the IC nearly at the same moment .

6.1 Place and route

The place and route of the chip has been done at IMEC (Leuven, Belgium) using the Aquarius XO Avant! tool. In fact the 0.35 μm Alcatel Mietec design kit requires the Synopsys software as the front-end tool for the synthesis and optimisation of the VHDL code and Avant! as the back-end one, for the final place and route. Since a license of the Avant! software has a prohibitive cost for the Universities or small industries some software-houses and microelectronics companies have bought a license of it and make the layout job for others with a certain price per day of work. We had this job done at IMEC, which is one of the microelectronics groups within Europractice that works as an intermediary between small design groups and the IC foundries providing both software and design kits. This job took 5 days for the layout design itself, then 3 days for the final layout verification using the Dracula tools. When a preliminary version of the layout was finished IMEC sent us a file with all the post-layout delays due to the interconnections. We then performed the post-layout simulations and verified that the chip works properly at the target frequency of 133 MHz.

One of the most effective features of the Avant! software concerns the clock tree synthesis: once decided the number of levels of buffers a clock tree is automatically synthesised by Avant! without having to route it manually as in other older tools. The skew of the clock net generated is 12 ps, that is completely neglectable at out target frequency. The final area of the chip is about 3 mm^2 versus the 14 mm^2 of the older one in 0.7 μm ES2 technology. The chip has already been submitted to the Alcatel Mietec foundry for the December 1998 run and the first prototypes will be ready for the beginning of April 1999. In figure 8 a picture of the chip layout is reported. As you can see the 3 memory blocks have been put on the left side of the die area. The upper block is the rule memory, while the others are the look-up table containing information of the input MFs. All

the 1700 standard cells have been put in a rectangular region on the right of the chip: they are surrounded by a double ring for power and ground nets. The clock net has been automatically traced by Avant! with a tree-shape structure in order to minimise the skew.

5. The HW and the SW developed to run the fuzzy processor

The VLSI fuzzy processor will be assembled on an printed circuit board for loading and running the fuzzy system. The fuzzy system can be loaded by means either a serial pin or the input data pins. The fuzzy chip assembled on the above printed board will be programmed via a multi I/O commercial board linked to a PC. In this way it is possible to load any fuzzy system into the fuzzy processor memories according to the handshake signal templates. In addition, by means of the same board, it is possible to run the fuzzy processor just by loading an input data set from the PC; otherwise, once a fuzzy system has been loaded, the input data set can also come from an external device. The printed board, where the fuzzy processor is going to be implemented, also has a clock signal and a battery for holding the fuzzy chip memory alive also when not running. Therefore the fuzzy chip can work off line to the PC to control the external device. Software programming tools have been developed running under Windows 95 with a friendly graphic interface:

- to edit the input data, the MFs and the rules;
- to automatically convert any fuzzy system in a new one where all the rules are present;
- to run the fuzzy system;
- to drive the fuzzy processor simulator.

The fuzzy simulator allows you to develop and test the fuzzy system, you are working with, without the board linked to the PC. It can generate also all the intermediate vectors related to each pipeline stage in order to test the design in the Synopsys environment and then to test the chip by the Tektronics ASIC tester when it comes back from the foundry. A warning is generated if errors are made. Debugging facilities are supplied: for example you can easily run the same set of data and MFs with different set of rules both with the chip simulator and with the chip itself. This software, according to the requested tasks, runs in two different modes: compiler mode and debugger one. In the compiler mode it works like an ordinary compiler just used to load properly the fuzzy chip memories while, in the debugger mode, a verbose list of all the intermediate vectors, which are involved in the inference process, is printed to the standard output.

8. Conclusions

We have been able to reach a clock frequency not higher than 133 MHz because the RAM megacells available at present are limited at this frequency. The two input fuzzy chip previously realised with the 0.7 μm technology has on area of 14 mm^2 while with 0.35 μm technology has about 3 mm^2 area. Moreover we have increased the throughput from 80 ns to 30 ns. This new chip will be assembled on a printed board and then can be linked to a PC to develop and test a fuzzy system for a specific application. Then it can work off-line linked to the device to be controlled as in the board there will be a clock and a battery. A four input fuzzy chip has been constructed two years ago [7] in ES2 0.7 micron, we plan to redesign it using this new technology and we hope not to have the limitation of the RAM megacell speed in order to reach a higher speed.

Acknowledgements

We greatefully acknowledge the "Laboratorio di Tecnologie Subnucleari" at the Department of Physics (University of Bologna) for using its instruments, in particular for the ASIC Tektronics tester used to test the fuzzy processor.

References:

1. Pagni, Poluzzi, Lo Presti, Rizzotto "Automatic Syntesis Analysis Implementation of a Fuzzy Controller" *IEEE International Conference on Fuzzy Systems 1993,* San Francisco, pp. 105-110.
2. Neichfeld, Klinche, Menke, Nolles, Kunemund "A General Purpose Fuzzy Inference Process" *IEEE International Conference on Microelectronics for Neural Networks and Fuzzy Systems,* Turin 1994, pp 310-318.
3. M. Sugeno "*Industrial Applications of Fuzzy Control*" Elsevier Science Pub. Co. 1985 - see

also Fuzzy Logic Handbook for use with Matlab pp. 2-53.

4. Ikeda, Kisu, Hiramoto, Nakamura "A Fuzzy Inference Coprocessor Using a Flexible Active-Rule-Driven Architecture" - *IKE IEEE 1992*, pp 537-544.
5. F. Boschetti, A. Gabrielli, E. Gandolfi, M. Masetti, "Fuzzy Logic Oriented to Active Rule Selector and Membership Function Generator for High Speed Digital Fuzzy Microprocessor " - *World Congress on Neural Networks*, July 17-21 1995 Washington - Vol. 3 pp. 93 – 97
6. R. Caponetto, M. Lavagna, M. Lo Presti, A. Milazzotto "Genetic Algorithms and Neuro-Fuzzy Systems for Automatic Controller Design" *Proceedings of CIFT'95*, Trento Italy June 8-10 95, pp 38-47
7. A. Gabrielli, E. Gandolfi, "A fast digital fuzzy processor", *IEEE Micro*, Jan-Feb 1999, pp. 68-79
8. F. Boschetti, A. Gabrielli, E. Gandolfi, M. Masetti, M. Russo "Digital Membership Function Generators and No-Contribute Rule Eliminator for High Speed Fuzzy Architectures "- *World Congress on Neural Networks* July 17-21 1995 Washington - Vol. 2 pp. 625 - 629
9. D, Falchieri, A. Gabrielli, E. Gandolfi, M. Masetti "Design and Realization of a Two Input Fuzzy Chip Running at a Rate of 80 ns" *Annual IEEE Meeting of the North America Fuzzy Information Processing Society* Syracuse, September 21-24, pp. 329-335.

.