Fifth IFSA World Congress (1993), 1390-1393

A Fuzzy Controller Chip for Complex Real-time Applications

Herbert Eichfeld, Thomas Künemund

Siemens AG, Corporate Research and Development, Dept. ZFE BT ACM 2,
Otto-Hahn-Ring 6, 8000 Munich 83, Germany

Abstract - **An 8b Fuzzy Coprocessor (FC) is presented that has eight programmable fuzzy algorithms and up to 256 inputs, 64 outputs and 16,384 rules. The 6.4mm^2 chip fabricated in 1.0µm CMOS technology can be used as a stand-alone device or as a macrocell for microcontrollers. Operating at 20MHz crystal frequency, it has a peak performance of 7.9M rules/s.**

Perspectives of future FC generations are also outlined, including a 12-16b resolution, additional fuzzy set operations, and optimized inference and defuzzification strategies.

I. INTRODUCTION

The Fuzzy Coprocessor (FC) chip presented in this article supports a host with fuzzy control computing.

Fuzzy Control applies the concepts of Fuzzy Set Theory to perform a control strategy based on empirical knowledge and expressed as a linguistic protocol in terms of if-then rules [1-3]. The fuzzy control algorithms are usually implemented with software running on standard hardware. However, if the system under control is complex and demands many control rules, or if it is fast and demands real-time control, special hardware is required. All previously introduced fuzzy hardware [4-7] suffers from a large silicon area. The FC, however, is small enough to be integrated as an on-chip microcontroller module. It is of particular interest for automotive and machine tool applications.

In sections II and III, the main features of the FC are listed and its operation is described. Architecture and memory organization are explained in sections IV and V. Finally, an outline is given of future FC generations.

II. FEATURES

The performance of the FC is given by

$$\mathbf{t = [\ (nr \cdot nc + 2^{ro} + ro + 1)\ tc + thf\ ]\ no,} \qquad (1)$$

where **t** is the computation time from initializing the FC until the output is calculated; **nr** is the number of if-then rules. The number of input read cycles **nc** is a function of the number of inputs **ni**:

$$\mathbf{nc = \ < ni\ /\ 4 >} \qquad (2)$$

with $< x > :=$ smallest natural number $\geq x$.

This means that the FC performs a 4-input parallel processing. The resolution of outputs is symbolized by **ro**, **tc** is the period of the system clock, **thf** the time for data transfer between host and FC. Finally, **no** stands for the number of outputs. The FC computes only one output at the same time. If there is a linguistic protocol with more than one output, the FC has to compute one after the other.

The term **ro +1** in (1) expresses the time demand for the division in the defuzzifier. The exponential term of **ro** signifies a true runtime integration for inference and defuzzifying.

In order to discuss the intrinsic FC performance, **thf** has to be left out of consideration, as it depends on the type of host applied. With an 8b output resolution **ro**, a system clock period **tc** of 100nsec, **no** set to one and **thf** set to zero, one gets:

$$\mathbf{t = (nr \cdot nc +\ 265) \cdot 100\ nsec.} \qquad (3)$$

Equation (3) is illustrated in Fig. 1. With 1000 rules and 4 inputs, for instance, the computation of the crisp output takes 126.5µsec. This results in a peak performance of 7.9 million rules per second.

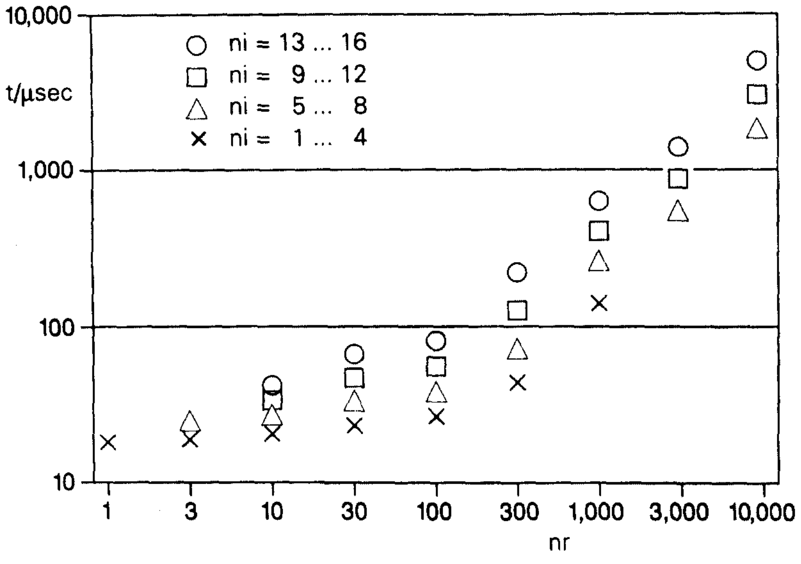

Fig. 1. Performance t as a function of the number of rules nr with the number of inputs ni as parameter.

Based on a VHDL description of the digital and all static logic design and applying logic synthesis, all circuits were mapped to standard cells in 1.0µm CMOS technology. The chip area is 6.4mm^2, including the on-chip RAM. An on-chip ROM stores the application-specific knowledge base. The crystal frequency can be up to 20MHz.

The FC is characterized by the following features:

- maximal number of inputs	256
- maximal number of outputs	64
- maximal number of rules	16,384
- Knowledge Base Memory (KBM)	
internal (max.)	64kB
external (max.)	64kB
- number of fuzzy algorithms	8
- resolution of I/O	8b

III. Operation

The host processor of the FC may be any microcontroller that can handle an 8b address/data bus. The first action in each control cycle is to address the FC control register with the start information. Secondly, the FC expects entry of digitized inputs one by one into its 8b data register. As soon as the crisp output is computed, the FC announces this with an IOR flag (Input Output Requested) in the control register pollable by the host, or with an interrupt signal. The host finally takes the crisp output from the FC data register. After that, another control cycle may start.

In case of more than 4 inputs, the first four are processed by the FC until it creates another IOR flag to request more inputs. The FC continues with inference calculations only after all inputs are sent by the host.

In case of more than 256 rules, a link of two or more knowledge bases (KB) is made. This offers the advantage that a large number of rules can be processed: linking all 64 KBs results in a maximal rule number of 64 · 256 = 16,384. In addition, the rules can be grouped in parts of equal numbers of input read cycles **nc**. Each group can be defined as a single KB which is linked to the others. This method of rule grouping allows memory saving and performance enhancement.

IV. Architecture

The functional blocks of the FC are drawn in Fig. 2. The data for the FC control register as well as the inputs are driven by the host via the host bus. The start information and the 8b inputs is sent via the internal data bus to the KBM interface which handles the communication with the KBM. There are two possible KBM implementations: on-chip ROM or off-chip memory devices, which may be of any type (SRAM, EEPROM,..).

The fuzzifier stores the values of the hit input membership functions (IMF). In the rule decoder, the linguistic values of the actual inputs are compared with the if-then rules to determine the active rules which are subsequently evaluated in the rule evaluator: first, the MIN-operator is applied to compute the fulfillment value of each active rule. After that, aggregation of the fulfill-

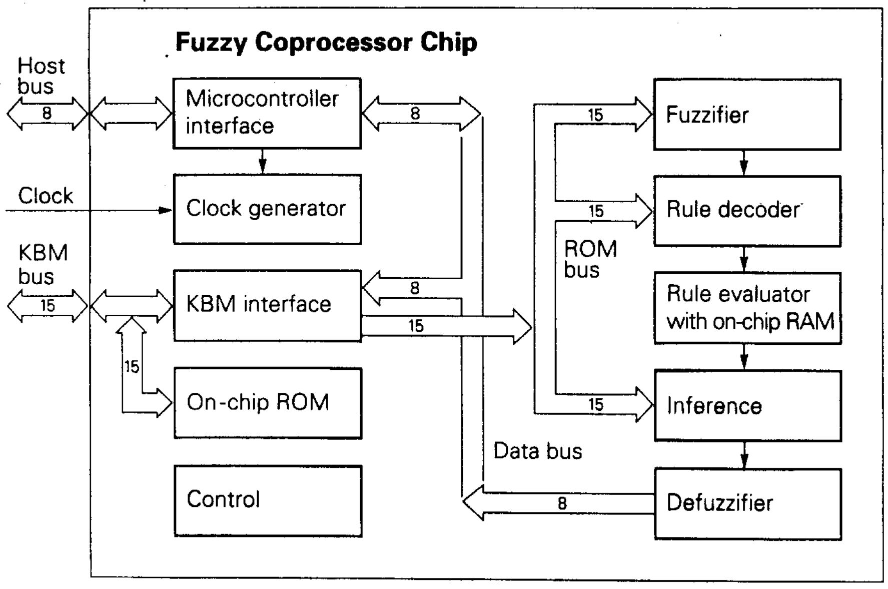

Fig. 2. Architecture of the Fuzzy Coprocessor

ment values is performed with the MAX operation or with the Bounded Sum (BSUM) in order to obtain the weights for each output membership function (OMF). That is, in case of BSUM all relevant rules contribute to the respective OMF weights, whereas for MAX only the maximum of the fulfillment values is taken into account. As one rule can connect up to 256 inputs, an on-chip RAM storing intermediate results is needed in the rule evaluator block.

The OMF from the KBM and their weights from the rule evaluator are the inputs for the inference unit to calculate the possibility distribution of the output. This fuzzy output is computed using MAX- or BSUM-operation of the weighted OMF. Here, BSUM means that the overlap of adjacent OMFs enters into the fuzzy output, whereas for MAX it is left out of consideration.

The last step is converting the fuzzy output into a crisp one. The defuzzification is implemented as a Center of Gravity (CoG) or a Mean of Maxima (MoM).

Thus, the user can choose between eight hardware-implemented fuzzy algorithms: MAX or BSUM in the rule evaluator, MAX or BSUM in the inference and CoG or MoM in the defuzzifier can be combined to eight program-selectable alternatives.

V. Memory

The KBM stores four types of information: Knowledge Base Descriptors (KBD), Input Membership Functions (IMF), Output Membership Functions (OMF) and Sets of Rules (SR).

The KBDs consist of four 15b words for control information and start addresses for IMF, OMF and SR.

The IMF and the OMF are stored in a look-up table in a format explained in [8]. Thus, every MF shape is programmable. However, the grade of overlap is set to a maximum of two. For the IMF, up to seven linguistic values, for the OMF, up to eight can be defined.

The SRs are if-then rules with a variable number of AND-connected inputs and one output. Each input is represented by a number signifying one of its linguistic values. For more than four inputs, the SRs have to be split into segments with four inputs.

After each control cycle, the KBM may move between on-chip and off-chip memory. Thus, the memory size can be easily extended.

The memory demand **m** is expressed by

$$\mathbf{m = 60 \cdot nc \cdot no + 15 \cdot nr \cdot nc \cdot no + 15 \cdot (2^{ri} \cdot ni + 2^{ro} \cdot no)}. \quad (4)$$

The first term stands for the memory demand of the KBDs, the second for that of the rules and the last for that of the membership functions. An example of memory organization can be found in [9].

Memory capacity can be saved in all cases where there are equal IMFs or OMFs or SRs in different KBs because the KBD allows memory sharing of different information. Another way of using this feature is to adapt the KB efficiently to slowly varying system characteristics if only parts of a KB are affected. A "degradation age", for example, which can be evaluated by the host, also could be used to control the choice of the appropriate KB.

VI. PERSPECTIVES

For future FC generations, two lines of further development have been envisaged: first, an even faster and more compact 8b design, and second, an extended version with 12-16b resolution of in/outputs.

A look at (1) shows that for small numbers of inputs **ni** and rules **nr**, the performance is dominated by the term $2^{ro} = 256$: the entire output domain is taken into account, irrespective of values for which the fuzzy output is zero. To be able to skip these regions not contributing to the calculation of the crisp output, additional circuits have been developed. Although all characteristics of a true runtime integration remain preserved, an acceleration up to a factor of four can be achieved.
On the other side, for large **ni** and **nr**, the major part of the computation time is needed for rule decoding and evaluation. So far, the antecedent part of the rules means an AND-operation on fuzzy sets (see V), i.e. expressing, for example, a NOT-operation results in a comparatively large number of rules. Therefore, hardware representations of additional fuzzy set operations would reduce **nr** significantly and thereby lead to a relative gain in performance and memory space strongly increasing with **ni**. As an example, for rule bases used in practice and connecting five to seven inputs with three to five linguistic values each, an improvement by at least a factor of five could be achieved.

Due to the 64 different KBs and by clever programming the above-described 8b FC, a limited number of inputs with higher resolution can also be processed, e.g. two inputs with 11b resolution each. In addition to that, by employing several chips in a parallel or hierarchical set-up, an even higher resolution and/or input number could be handled. But, even if the performance were to remain reasonable, such solutions will be of interest only for a limited number of applications because of the exponentially increasing silicon demand.
From (4) it is obvious that in high resolution FCs it is no longer possible to process membership functions of arbitrary contour: a restriction is imperative, e.g. to a trapezoidal shape. This was done in [10] where a special-purpose fuzzy microprocessor is described.
However, additional effort will be necessary to arrive at a sufficiently fast 12-16b FC which is also small enough to be integrated at low cost on a standard microcontroller. This means an area-optimized IMF calculation and a pipelined operation of fuzzifying, rule decoding and rule evaluation. A method for screening of not-active rules would also be useful which allows for significant performance enhancement in case of large **ni** and **nr**. Finally, highly elaborate defuzzification strategies are needed, consisting of preprocessing the OMF as far as possible in order to avoid the performance penalty of an actually performed high resolution runtime integration without destroying its information content.

ACKNOWLEDGEMENT

The authors would like to thank M. Klimke, S. Lewalter, J. Nolles and their colleagues of the Semiconductor Group for their fruitful and efficient cooperation. They also wish to thank D. Schmitt-Landsiedel and H. Herbst for encouragement and support throughout the work.

REFERENCES

[1] L.A. Zadeh,"Fuzzy Sets", *Informat. Control*, vol.8, pp. 338-53, 1965.

[2] E. H. Mamdani, S. Assilian, "A Case Study On The Application Of Fuzzy Set Theory To Automatic Control", *Proc. IFAC Stochastic Control Symp.*, Budapest 1974, pp. 643-9.

[3] C.C. Lee,"Fuzzy Logic in Control Systems: Fuzzy Logic Controller", *IEEE Trans. Syst. Man Cybern.*, vol. SMC-20, no. 2, March/April 1990,"- Part I", pp. 404 - 18,"- Part II", pp. 419 - 35.

[4] T. Yamakawa, "High-Speed Fuzzy Controller Hardware System: The Mega-FIPS Machine", *Information Sciences* 45, 113 - 128 (1988).

[5] R. J. Corder, "A High-Speed Fuzzy Processor", *Proc. of the 3rd IFSA World Congress1989*, pp. 379 - 81.

[6] H. Watanabe et al., "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture", *IEEE J. Solid-State Circuits*, vol. SC-25, No. 2, April 1990, pp. 376-82.

[7] H. Ikeda et al., "A Fuzzy Inference Coprocessor Using a Flexible Active-Rule-Driven Architecture", *Proc. of the 1st Int. Conf. on Fuzzy Systems FUZZ-IEEE'92*, pp. 537 - 44.

[8] H. Eichfeld et al., "Architecture of a CMOS Fuzzy Logic Controller with optimized memory organization and operator design", *Proc. of the 1st Int. Conf. on Fuzzy Systems FUZZ-IEEE'92*, pp. 1317 - 23.

[9] H. Eichfeld, T. Künemund, M. Klimke, "An 8b Fuzzy Coprocessor for Fuzzy Contol", *ISSCC Dig. Tech. Pap.*, Feb.1993, pp. 180-181.

[10] K. Nakamura et. al., "A 12b Resolution 200 kFLIPS Fuzzy Inference Processor", *ISSCC Dig. Tech. Pap.*, Feb.1993, pp 182-183.