

Optimized Fuzzy Controller Architecture for Field Programmable Gate Arrays

Hartmut Surmann, Ansgar Ungering and Karl Goser

University of Dortmund, Faculty of Electrical Engineering, *
44221 Dortmund, Germany,
E-mail: surmann@luzi.e-technik.uni-dortmund.de

Abstract. This paper describes an optimized fuzzy controller (FC) architecture and its realization with field programmable gate arrays (FPGAs). In consideration of data dependencies and minor user restrictions within the definition of fuzzy rules (FRs), it is possible to develop a high speed FPGA architecture. A prototype of the FC operates at 5MHz and needs 50 μ s operation time (8 bit resolution) independent of the number of inputs/outputs with 256 fuzzy rules. A pipeline architecture is used to achieve a high processing speed.

1 Introduction

Since Mamdani's work [1] on fuzzy control, which was motivated by Zadeh's approach to inexact reasoning, a lot of work has been reported in this research field so far. The basic idea of this approach was to incorporate the control know-how of a skilled human operator by fuzzy sets and fuzzy rules. The FRs are combined by the fuzzy implication and the compositional rule of inference.

The difficulty of the control know-how is due to their non-linear, time varying behaviour and the poor quality of the available measurements. Fuzzy logic replaces "true" and "false" with continuous membership values ranging from zero to one, which mirror natural language concepts. This allows to process linguistic concepts (adjectives, adverbs) like "small", "big", "near", or "approximately" in the control system. The main advance is to control processes which are too complex to be mathematically modelled in real time.

In the first fuzzy applications FC's are not optimized implemented on standard microprocessors which are flexible, but these first implementations prevent high speed computing. The first approach to get higher performance was to build special microprocessors (RISCs, like FC-110, 80C166) with a special fuzzy instruction set. A much cheaper and faster approach in terms of processing and implementation time is to consider data dependencies and to implement an optimized controller algorithm on a fast 32-bit RISC or standard microprocessor [2]. In addition to the advantages of a FC a very high processing speed and low hardware costs are necessary to achieve a wide acceptance.

* This work is supported by the VW-Stiftung, Hannover

In this paper we present an architecture of a general purpose FC and a prototype realization with 256 fuzzy rules, 4 inputs variables and 1 output variable (8 bit resolution). The regular and modular structure of the controller also motivates a VLSI implementation [3, 4, 5, 6, 7, 8].

2 Basic terms of Fuzzy Controller

The FC algorithm is based on the generalized modus ponens inference rule [3]:

$$\begin{array}{l} \text{Premise: } \quad \text{A is true} \\ \text{Implication: If A then B} \\ \hline \text{Conclusion: B is true} \end{array}$$

The “crisp” propositions A and B are replaced by fuzzy functions. Fuzzy functions characterise and define fuzzy sets through $\mu_i : U \rightarrow [0, 1]$ with $e \mapsto \mu_i(e)$, so $i = \{(e, \mu_i(e)) | e \in U, \mu_i(e)\}$. For fuzzy sets Zadeh [9] defines for $e \in U$ three important fuzzy operations:

$$\begin{array}{lll} \text{Intersection } C = A \cap B & \text{Union } C = A \cup B & \text{Complement } \bar{A} \\ \mu_C(e) = \min(\mu_A(e), \mu_B(e)) & \mu_C(e) = \max(\mu_A(e), \mu_B(e)) & \mu_{\neg A}(e) = 1 - \mu_{\bar{A}}(e) \end{array}$$

A, B and C are fuzzy sets and U is the universe of discourse for e . These fundamental operations together with the set $[0,1]$ forms a fuzzy algebra, so that any logic function can be build. Instead of $\mu_A(e)$ we only write A to denote the fuzzy set A. The Boolean algebra is a subset of the fuzzy algebra and can be implemented by replacing continuous functions with unit pulses. In difference to a conventional knowledge based system, the premise of the rule is a value in $[0,1]$ instead of $\{0,1\}$. The example [10] in Fig. 1 introduces the basic FC algorithm. It shows three simple rules for charging batteries with two inputs dU (gradient of voltage), T (temperature) and one output I (current).

- R1: IF dU is negativ and T is normal THEN I is low
- R2: IF dU is positiv and T is high THEN I is low
- R3: IF dU is positiv and T is normal THEN I is high

The fuzzified inputs dU and T are simultaneously switched to all the rules to be compared with the stored premises (IF parts). Now the truth values $\alpha_i^{dU}, \alpha_i^T$ for every subpremise are calculated by:

$$\begin{array}{l} \alpha_i^{dU} = \mu_{A_i}(dU) \quad , \text{ for } A_i \in \{\text{negativ, positiv}\} \\ \alpha_i^T = \mu_{B_i}(T) \quad , \text{ for } B_i \in \{\text{normal, high}\}, \quad \text{for } i = 1 \dots 3 \end{array} \quad (1)$$

$\alpha_1^{dU} = 0.0$ indicates that the input completely mismatches with the stored subpremise which leads to a complete noncontribution of rule 1 to the output. $\alpha_2^{dU} = 0.85$ and $\alpha_2^T = 0.65$ in rule 2 generates a rule matching or truth value of $\omega_i = 0.65$, because the fuzzy logic conjunction “and” is interpreted as the minimum of α_i^{dU} and α_i^T ($\omega_i = \min(\alpha_i^{dU}, \alpha_i^T)$). The conclusion of each rule is

$$I'_i = \{\min(\omega_i, x) \mid x \in I_i\}, \quad \text{for } i = 1 \dots 3 \quad (2)$$

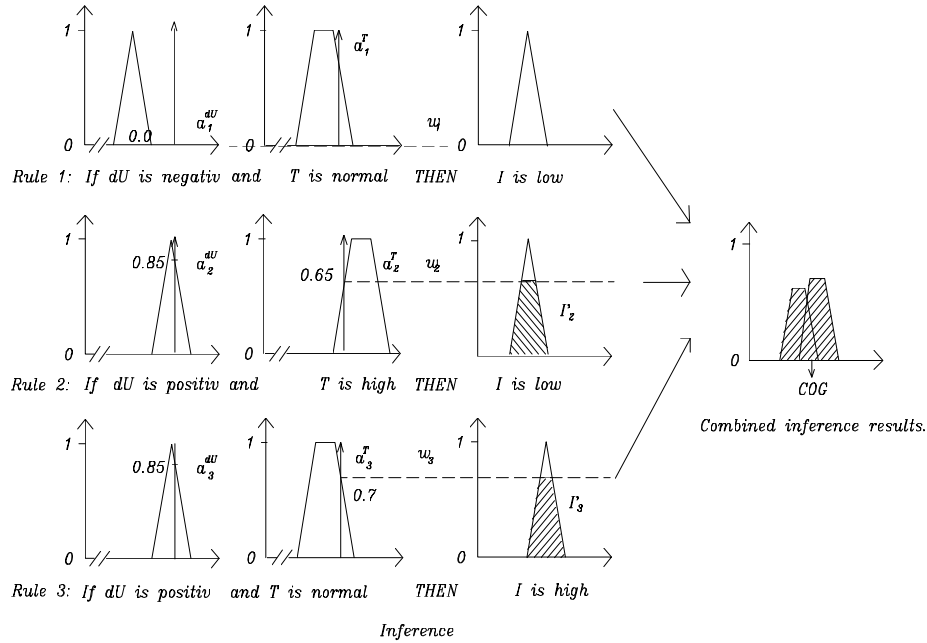


Fig. 1. Fuzzy controller algorithm

and represents the THEN part of each rule. The fuzzy result function I' is the unification of all subresults I'_i and is calculated by $I' = \bigcup I'_i$, for $i = 1 \dots 3$. In most applications the output values are “crisp” numbers (unit pulses), which are accomplished by calculating the center of gravity (COG) of the resulting fuzzy function I' :

$$COG_I = \frac{\int e \times I'(e) de}{\int I'(e) de} \quad (3)$$

The described FC with binary input and output values is called BIOFAMs (Binary Input-Output Fuzzy Associative Memory) [11] or MIN-MAX algorithm [1] with the COG used as defuzzification method.

3 The architecture

A typical FC consists of four different units:

- the fuzzy rule base,
- the inference unit,
- the fuzzifier unit,
- the defuzzification unit.

Only parallel architectures can achieve the realtime requirements. First digital implementations [3] have a high flexibility but need a large amount of chip area because they store the same membership functions several times. After

an analyse of the applications, new architectures [5, 8] restrict the degree of overlab and the number of the membership functions. Figure 2 shows the basic structure of the architecture. The FC has a three stage pipeline and two different

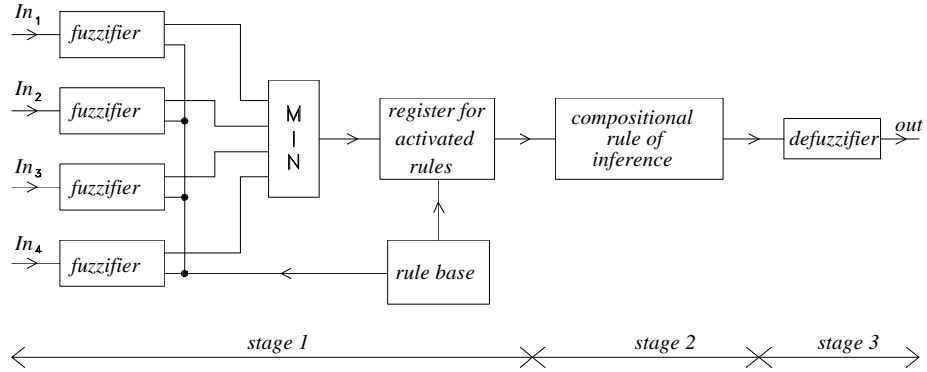


Fig. 2. Pipeline architecture of the fuzzy controller

clock cycles (internal main clock Φ , i.e. 5 MHz and external $\phi = \Phi/256$) which are not shown in Fig. 2. The main clock Φ controls the data flow between the pipeline blocks. The first block process one fuzzy rule each clock cycle ϕ , so that this architecture can handle 256 rules. The results of the premise calculation are stored in a special register block and will be transferred in the next block with the next main clock cycle. There, the output membership functions are limited and the composition is evaluated. The last block (defuzzifier) calculates the “crisp” output value.

3.1 The rule base

The transfer function of a FC is supplied by the definitions in the fuzzy rule base. This rule base connects the membership functions with the “crisp” input values and defines which membership function is activated. The storage capacity S_R for n input and m output variables with k membership functions per input/output variable is $S_R = [ld(k)](m + n)$ (Tab. 1).

Table 1. Binary code (b) of a fuzzy rule base (a), $n=3$, $m=1$, $k=8$.

Rule 1: if A' is A3 and B' is B4 and C' is C7 then X4	011 100 111 100
Rule 2: if A' is A2 and B' is B4 and C' is C3 then X6	010 100 011 110
...	...
(a)	(b)

3.2 The Fuzzifier unit

In the fuzzifier unit the inputs are compared with the stored premises (IF parts) and the truth values are calculated for every subpremise. A simple and fast method to store the premises (membership functions) is to use a RAM. Therefore, the input value is the address of the RAM blocks. Eichfeld et al. [5] described an optimized memory organisation in which the membership functions get a binary number and the overlapped membership functions are stored in different RAMs (Fig. 3). Three different memory blocks (two for the membership functions and one for the numbers) are required if only two neighbored membership functions overlap. A higher overlap degree requires one more memory block.

The FPGA implementation allows the definition of 8 different membership functions for each input/output variable with a resolution of 8 bit, which requires a storage capacity of $SR = 2 * 256 * 8 + 2 * 256 * 2 = 5120 \text{ bit} = 640 \text{ bytes}$ per input/output variable. If a higher resolution or more flexibility is demanded, e.g. for fuzzy processors, then a membership function generator [8] can be used.

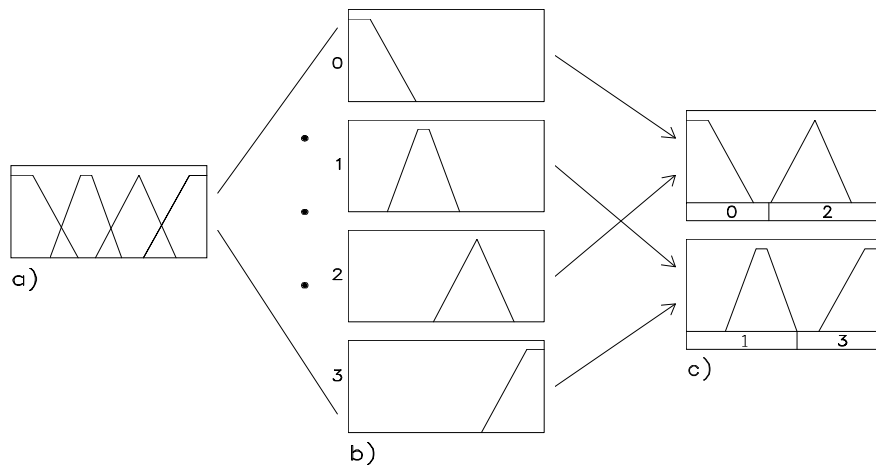


Fig. 3. Compressed storage of neighbored membership functions

Figure 4 describes the calculation of the truth value with an optimized memory organization. The input value (183) is the address for the two memory blocks. There, the membership functions (MFs) 5 and 6 are activated with $\alpha_5 = 204$ and $\alpha_6 = 128$. MF 5 is selected in the subpremise of the rule base so that the multiplexer switches α_5 to 204. For each other MF a zero is processed by the multiplexer. The memory organisation with odd and even MF numbers saves an extra bit while the LSB equals one for odd binary numbers, so that instead of 3 bit only 2 bits are required.

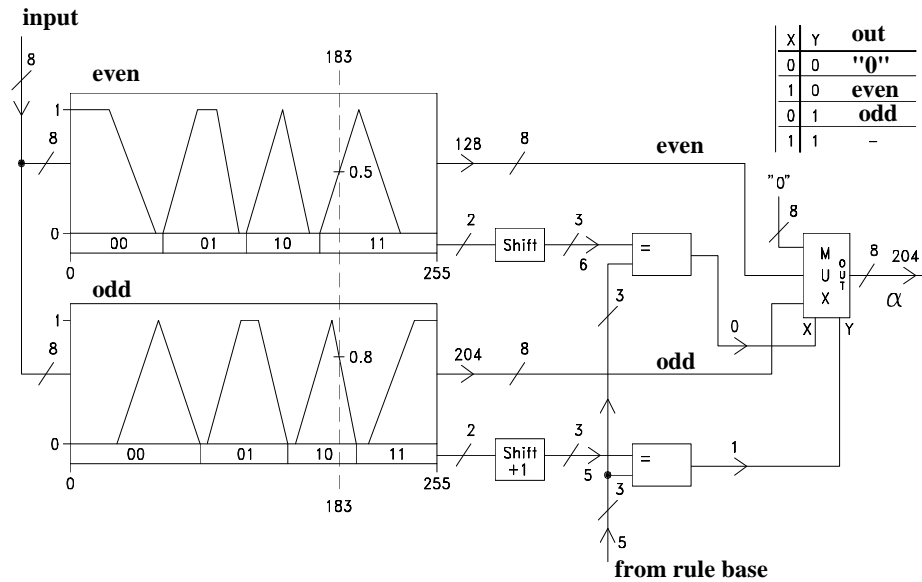


Fig. 4. Calculation of the truth value of the subpremise

3.3 Calculation of activated rules

As mentioned in chapter 2 the fuzzy logic conjunction "and" is interpreted as the minimum operation of the subpremises truth values α^i . If the truth value ω of a fuzzy rule is greater than zero then the rule is activated and delivers a contribution to the output value. The truth value ω_{act} for a fuzzy rule is stored in the register block with the MF number (Fig. 5) if ω_{act} is greater than the previous truth value ω_{pre} for the MF (maximum operation). Depending on the MF number, the multiplexer connects the truth value ω_{pre} back to the maximum circuit. After all rules are evaluated, the values of the first register block Reg0 ... Reg7 are transferred in the second register block Reg0' ... Reg7' of the second pipeline stage.

3.4 Composition rule of inference

The MFs for the output variables are organized similar to the MF for the input variables, so that the same memory organization is used. The inference algorithm limits the output MF with the truth value ω ($I'_i = \{min(\omega_i, x) \mid x \in I_i\}$). Therefore, all membership values have to be computed to calculate the resulting MF (Fig. 6). An 8 bit counter generates the addresses for the odd and even memory blocks. Together with the membership value the binary number of the membership function is selected to address the register block with the truth value ω . The minimum circuit limits the output membership function and the maximum circuit computes the compositional rule of inference. Different fuzzy

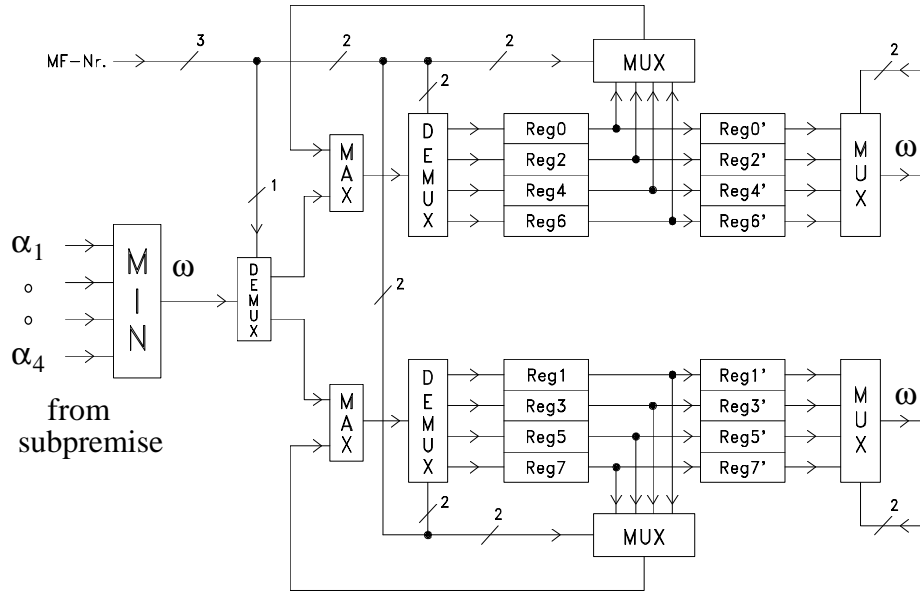


Fig. 5. Fuzzy "and" and computation of activated rules

controllers can be connected over the CAS_{in} and CAS_{out} lines to build complex systems.

3.5 Defuzzifier

To evaluate a "crisp" value (unit pulse) from the output membership function the COG has to be calculated (chap. 2). Therefore summation, multiplication and division operations have to be carried out. The concept of the repeated adder of Watanabe et al. [3] economises the multiplication operation (Fig. 7). The denominator is simply the summation of the data stream from the inference stage. Since the numerator can be computed by repeatedly adding the denominator, all summation operations can be done in the second pipeline stage. The division operation occurs in the third pipeline stage after the calculation of the numerator and denominator.

3.6 FPGA implementation

A prototype of the architecture with 4 inputs, 1 output and 256 rules is implemented on 2 FPGA (XC3090-100 PG84C, 320 CLBs [12]), both using about 80% of the CLBs. The rule base and membership functions are stored in external RAMs (8 bit, 20ns). The fuzzy conjunction and the computation of activated rules (Fig. 5), together with minimum and maximum circuits (Fig. 6), is placed on the first FPGA. The repeated adder is placed on the second FPGA together

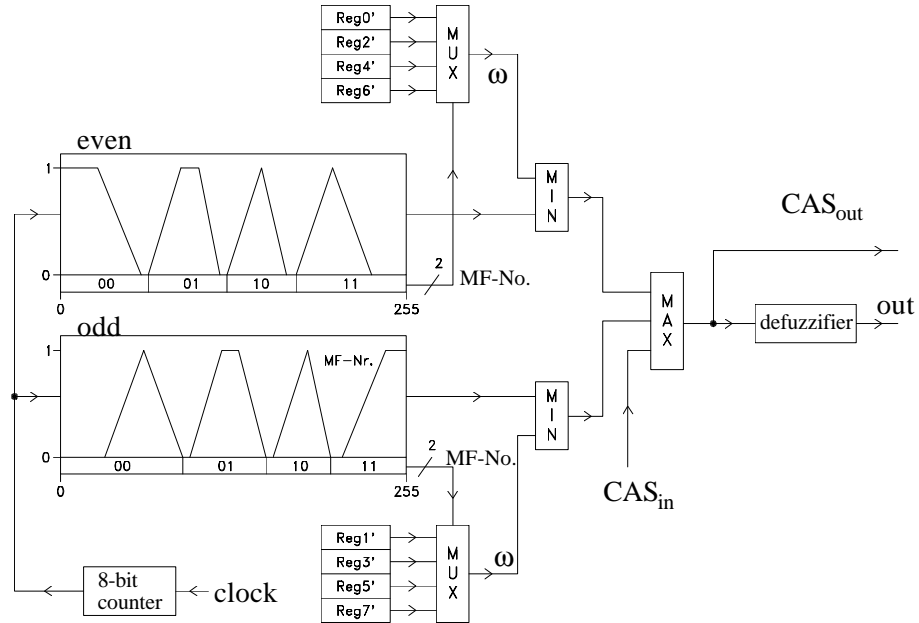


Fig. 6. Compositional rule of inference

with the divider for the defuzzification. The place and route is done half automatic which means that we placed the critical blocks and wired the critical paths by hand.

The third generation of LCA devices like the XILINX XC 4000 family can be considerably increase the performance of the described FC. By using the new devices the number of chips will also be reduced, so that the controller can easily be placed on a single FPGA.

4 Conclusion

We have introduced an optimized fuzzy controller (FC) architecture and its realization with field programmable gate arrays (FPGAs). In consideration of data dependencies and minor user restrictions by the definition of fuzzy rules (FR), it was possible to develop a high speed FPGA architecture. A prototype of the FC operates at 5MHz and needs 50 μ s (8 bit resolution) operation time independent of the number of inputs/outputs with 256 fuzzy rules. Future research will focus on microelectronic VLSI realization of adaptive fuzzy controllers [13] as well as different applications.

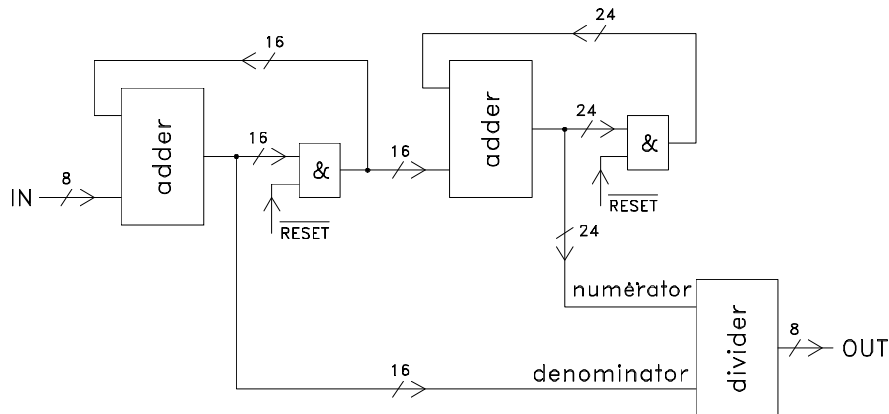


Fig. 7. Repeated adder [3]

References

1. Mamdani, E.H. : Application of fuzzy algorithms for the control of a dynamic plant. Proc. IEEE 121, No.12, (1974) 1585-1588
2. Surmann, H., Heesche, K., Hoh, H., Goser, K., Rudolf, R.: Entwicklungsumgebung für Fuzzy-Controller mit neuronaler Komponente. VDE-Fachtagung Technische Anwendungen von Fuzzy-Systemen, Dortmund, (12/13.Nov. 1992) 288-297
3. Watanabe, H., Dettloff W.D., Yount, K. : A VLSI fuzzy logic controller with reconfigurable, cascable architecture. IEEE journal of solid state circuits, Vol.25, No.2, (1990) 376-382
4. Ungerling, A., Qubbaj B., Goser, K.: Geschwindigkeits- und speicheroptimierte VLSI-Architektur für Fuzzy-Controller. VDE-Fachtagung Technische Anwendungen von Fuzzy-Systemen, Dortmund, (12/13. Nov. 1992) 317-325
5. Eichfeld, H., Löhner, M., Müller, M.: Architecture of a Fuzzy Logic Controller with optimized memory organisation and operator design. Int. Conf. on Fuzzy Systems, FUZZ-IEEE '92, San Diego, (March 8-12, 1992)
6. Surmann, H., Tauber, T., Ungerling, A., Goser, K.: Architecture of a Fuzzy Controller based on Field Programmable Gate Arrays. 2nd International Workshop on Field-Programmable Logic and Applications, Wien, 31. (Aug. - 2. Sept. 1992)
7. Surmann, H., Ungerling, A., Goser, K.: Fuzzy Controller mit VLSI-Pipeline-Architektur für hohe Datenraten. ITG-Fachbericht 119, Mikroelektronik für die Informationstechnik, Stuttgart, (4-6 März 1992) 195-200
8. Ungerling, A., Roer, P., Surmann, H., Daszkiewicz, D., Goser, K.: Architekturkonzept eines Fuzzy-RISC-Prozessors mit optimierten Speicherbedarf. ITG/GME/GI-Fachtagung, Rechnergestützter Entwurf und Architektur mikroelektronischer Systeme, Darmstadt, (23/24. Nov. 1992) 229-238
9. Zadeh, L.A.: Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. IEEE Transactions on systems, man and cybernetics, Vol.SMC-3, No.1, (Jan. 1973) 28-32
10. Surmann, H., Flinspach, G.: Fuzzy-Controller gesteuertes Schnell-Ladeverfahren für NiCd-Akkumulatoren. VDE-Fachtagung Technische Anwendungen von Fuzzy-Systemen, Dortmund (12/13. Nov. 1992) 159-168

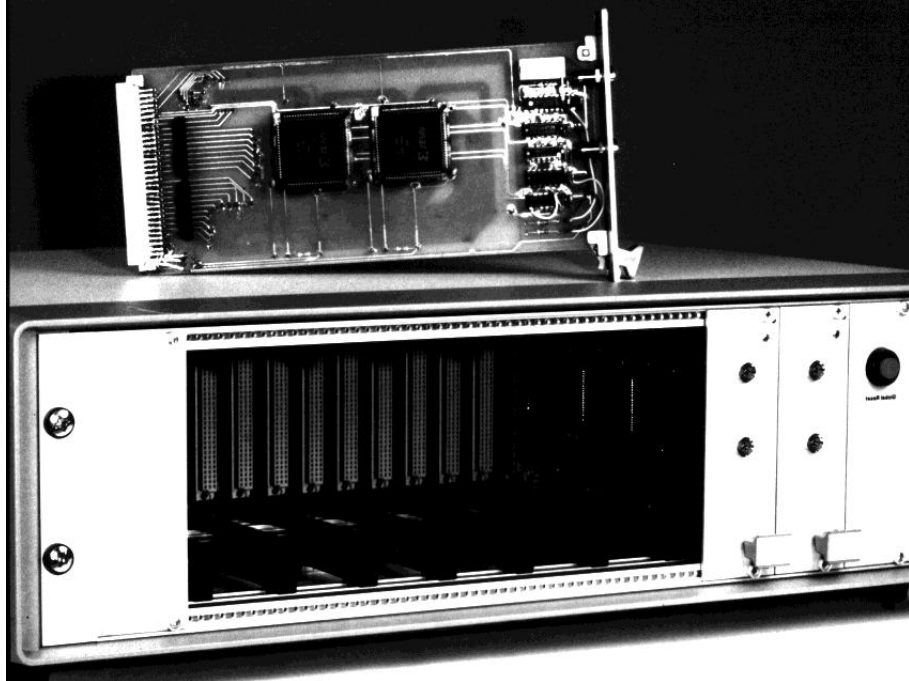


Fig. 8. Development board

11. Kosko, B.: Neural networks and fuzzy systems - A dynamical systems approach to machine intelligence. Prentice-Hall (1992)
12. XILINX Inc.: The Programmable Gate Array Data Book. Users Guide and Tutorial Book. San Jose / California, (1991)
13. Surmann, H., Möller, B., Goser, K.: A distributed self-organizing fuzzy rule-based system. Proceedings Neuro-Nimes 92 (1992) 187-194