The Method of Hardware Implementation of Fuzzy Systems on FPGA

Andrzej Przybyl[1(✉)] and Meng Joo Er[2]

[1] Institute of Computational Intelligence,
Częstochowa University of Technology, Częstochowa, Poland
andrzej.przybyl@iisi.pcz.pl
[2] School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore, Singapore
emjer@ntu.edu.sg

Abstract. In this paper a method of implementation of fuzzy system on FPGA devices is presented. The method applies to a class of fuzzy systems which are functionally equivalent to a radial basis function networks. In the paper the example fuzzy system was implemented on the FPGA device with the use of the proposed method. The results confirm a high performance of the obtained fuzzy system. This was achieved at a reasonable consumption of the hardware resources of the FPGA.

Keywords: Hardware implementation of fuzzy systems · FPGA · Radial basis function

1 Introduction

Computational intelligence methods (see e.g. [2–4,6–10,13–15,17,29–32,34,38–40,42,48–54,60–64]) offer suitable properties for modeling the nonlinear dynamics of various types of real objects. A different types of neural networks (see e.g. [16,55]) or fuzzy systems (see e.g. [18–27,41,43,56–59,68–74]) have a number of useful features such as the ability to approximate any continuous non-linearity or the ability to interpret the accumulated knowledge. However, from a practical point of view, the other features are also important. For example, the ability to implement in a hardware (e.g. FPGA) to obtain the operation model working in a real time. Moreover, the implementation should be relatively simple and economically justified.

In recent years, a large number of projects have used FPGAs to perform the control and modeling of dynamical systems. In many cases, these projects utilize neural networks [5], fuzzy systems or neuro-fuzzy systems [11,33]. However, in some cases, the degree of complexity of used algorithms is very high and the economy of this solution is questionable.

This is due to the fact that these algorithms are mainly based on arithmetic operations for floating point numbers. In particular floating-point operations such as divide numbers [37], exponential and trigonometric functions are characterized as they have the high complexity and low performance when they are

L. Rutkowski et al. (Eds.): ICAISC 2016, Part I, LNAI 9692, pp. 284–298, 2016.
DOI: 10.1007/978-3-319-39378-0_25

implemented on FPGA devices. FPGA hardware resources are not adapted to the efficient implementation of this type of calculations. FPGAs are well suited for the implementation of fixed-point calculations, such as addition, subtraction and multiplication. Implementation of complex arithmetic operations based on the floating-point numbers consumes a lot of resources of the FPGA hardware. For this reason, in most cases fully parallel implementation of floating-point calculations becomes economically unjustified.

In order to reduce the high consumption of resources a serial or semi-parallel data processing algorithms are used, including the recursive implementations [37]. In this case, the demand for hardware resources significantly decreases. However, computing efficiency drops significantly - which is an obvious drawback of such a solution. It should be noted that in some cases this approach is highly justified. For example, consider the control system whose duty cycle is limited by the limit frequency of operation of actuator, for example about 20 kHz. In this case, the hardware implementation of the complete control algorithm working with the cycle less than 50 µs is pointless, because the generated data are not used earlier than the mentioned time 50 µs elapses. It should be noted that there are a number of applications which drew significant benefits if the processing time is as short as possible. Examples are hardware emulators of various types of real objects used for hardware-in-the-loop (HIL) systems.

As noted in the work [46] there are existing commercial digital real-time HIL simulators that are characterized by 50 µs to 100 µs time steps and computational latency, and therefore they are not able to simulate accurately the very fast dynamics of power electronics systems. The authors suggest that simulation with a time step with value of 1 µ or less is much more appropriate solution. In order to obtain high processing speeds various techniques are used. They cover both the structure of the implemented algorithm and methods of their implementation.

The vast majority of practical implementation on FPGA widely use triangular or trapezoidal fuzzy sets. Such sets are easier to be realized in FPGA than the ones based on a Gaussian functions [1,47]. While many theoretically developed algorithms are based on Gaussian fuzzy sets, which sometimes are considered to be more appropriate to represent fuzzy knowledge. Moreover, if the input variables are represented by complementary membership functions of the fuzzy sets it follows another benefit, namely processing technique is applied only for activated rules. How was indicated in the paper [47], elimination of verification of the activation degree of all fuzzy rules allows to accelerate inference process. One of the possible techniques used in this field is the odd-even method [28].

The results presented in various papers show that in many cases relatively high processing speed is achieved, however, at the expense of low resolution of processed signals. This is due to the applied binary encoding using an average of 6 to 8 bits. Unfortunately, the specificity of many proposed solutions is that increasing resolution of processed words, eg. to 12-bits, causes a significant increase in the consumption of hardware resources.

In this paper we propose a new method for the implementation of fuzzy system on FPGA. This method offers good performance and accuracy with relatively low consumption of hardware resources.

This paper is organized into 4 sections. Section 2 contains an idea of designing the neuro-fuzzy structure to the limitations arising from the implementation in hardware in FPGA devices. Implementation results are presented in Sect. 3. Conclusions are drawn in Sect. 4.

2 The Method of Designing the Fuzzy Structure to the Limitations Arising from the Hardware Implementation

In this work will be considered systems using fuzzy rules of the following form

$$\textbf{IF } (x_1 \textit{ is } \overline{x_1^j}) \textbf{ AND } ... \textbf{ AND } (x_N \textit{ is } \overline{x_N}) \textbf{ THEN}(y \textit{ is } \overline{y}),$$

where x_i indicates the input to the system ($i = 1..N$), y is the output, $\overline{x}_i^j$ are input fuzzy sets for the j-th fuzzy rule ($j = 1..M$) and $\overline{y}^j$ are output fuzzy sets. In the considered systems Gaussian input fuzzy sets are used. The algebraic product is used as a T-norm operator. Rules consequents are a singleton type and the method of centre of gravity for singletons (COGS) is used for defuzzification. For the sake of clarity of description we will present a system with one output. It should be noted that such simplification does not constitute the loss of generality for the general idea presented in this paper.

According to the theory of fuzzy logic and common practice the implementation of fuzzy system is followed in three stages: 1. fuzzification, 2. inference, 3. defuzzification. However, because of the investigated class of fuzzy systems are functionally equivalent to a radial basis function networks [36] (it will be explained in detail in the later in the paper), in the current paper it is proposed a more appropriate method of hardware implementation.

The main features of the proposed method are: 1. operations are implemented in hardware based on fixed-point and simplified floating-point arithmetic, 2. fuzzification and inference is carried out together on the basis of functional similarity to radial basis function networks.

The proposed method is scalable and allows adjustment of the obtained processing speed and the use of hardware resources for a specific application. The next part of the work will present a detailed description of the proposed method of hardware implementation for the considered class of fuzzy structures.

2.1 The Method of Hardware Implementation of the Fuzzification and the Inference Processes

As pointed out in [66] and cited for this statement [35] the most important advantage of using fuzzy basis functions, rather than polynomials or radial basis functions, etc., is that a linguistic fuzzy IF-THEN rule is naturally related to a

fuzzy basis function. It should be noted that the way of the design of the system and its implementation not necessarily have to be identical. The design method should be intuitive to the man, while the implementation should be characterized by high efficiency and low cost. Therefore, let's look closer at the class of fuzzy systems presented in the previous section which are functionally equivalent to a radial basis function networks.

In the considered group of systems we assume that we are dealing with a Gaussian input fuzzy sets and every j-th rule uses of separate input fuzzy sets that are unshared with other rules. For each i-th input it exist a degree of membership to the i-th input fuzzy set of j-th fuzzy rule as follows:

$$\mu_i^j = \exp\left(-\left(\frac{x_i - \overline{x}_i^j}{\sigma_i^j}\right)^2\right), \tag{1}$$

where $\overline{x}_i^j$ and σ_i^j are center and width of input fuzzy set. If we use the product as the T-norm, then the degree of activity of the j-th rule is

$$\mu^j = \mu_1^j \cdot \mu_2^j \cdot \ldots \cdot \mu_N^j = exp\left(-\left(\frac{x_1 - \overline{x}_1^j}{\sigma_1^j}\right)^2 - \ldots - \left(\frac{x_n - \overline{x}_n^j}{\sigma_N^j}\right)^2\right). \tag{2}$$

Note that the action outlined above is similar to the way of calculating the values of the radial basis function of the following form

$$\mu^j = \exp\left(-\|\mathbf{x} - \overline{\mathbf{x}}^j\|^2\right), \tag{3}$$

in which the distance of the input $\mathbf{x}$ from the center of the radial fuzzy set $\overline{\mathbf{x}}^{\mathbf{j}}$ for j-th rule is defined as follows

$$\|\mathbf{x} - \overline{\mathbf{x}}^j\| = \sqrt{\left(\frac{x_1 - \overline{x}_1}{\sigma_1^j}\right)^2 + \ldots + \left(\frac{x_N - \overline{x}_N^j}{\sigma_N^j}\right)^2}. \tag{4}$$

This phenomenon has been observed and described in the work [36] as a functional equivalence between radial basis function networks and fuzzy inference systems. The specific form, for which each of the inputs has individually defined width σ_i^j of the set is called Hyper Radial Basis Function (HRBF) [45]. In the later part of this work the term fuzzy system (FS) refers to the category of systems that are functionally equivalent to a radial basis function networks.

The Eq. (4) can be rewritten as

$$\|\mathbf{x} - \overline{\mathbf{x}}^j\|^2 = w_0^j + w_{1,1}^j x_1 + w_{1,2}^j (x_1)^2 + \ldots + w_{N,1}^j x_N + w_{N,2}^j (x_N)^2, \tag{5}$$

where

$$w_0^j = \left(\frac{\overline{x}_1^j}{\sigma_1^j}\right)^2 + \ldots + \left(\frac{\overline{x}_N^j}{\sigma_N^j}\right)^2; w_{i,1}^j = -\frac{2\overline{x}_i^j}{\sigma_i^j}; w_{i,2}^j = \left(\frac{1}{\sigma_i^j}\right)^2. \tag{6}$$

The approach represented by formulas (3) and (5) offers noticeable benefits for the implementation, namely:

1. The Gaussian function is determined only once, in contrast to the classical approach, demanding the use of this function to determine the degree of membership of each input separately (1). The proposed solution is therefore beneficial because the Gaussian function is a troublesome operation in the implementation on FPGA.
2. All other actions necessary to determine the degree of activity of fuzzy rule are based on repetitive and simple activities such as multiplication and addition. These actions are easy to implement on FPGA, they are carried at high speed and consume relatively small hardware resources.

2.2 The Method of Implementation of Defuzzification Process

As it was mentioned earlier in the paper, the singleton membership functions with centers of $\overline{y}^j$ are used on the outputs of the rules. In the defuzzification stage the centre of gravity for singletons (7) is used because of the following features of the method: defuzzified values tend to move smoothly, have good sensitivity to change on inputs and are easy to calculate. According to the paper [12] the centre of gravity for singletons (COGSs) is the most realistic and widely used method of defuzzification in many applications.

$$y = \frac{\sum_{j=1}^{M} \mu^j \cdot \overline{y}^j}{\sum_{j=1}^{M} \mu^j} = \frac{n}{d} \tag{7}$$

However, from a practical point of view, it should be noted that this method is difficult to implement, because of used arithmetic division of real numbers. This operation can be avoided if fuzzy system is designed in such a way that the denominator in the formula (7) is equal to one, i.e. $d = 1$. This approach is very comfortable and quite often used in practice. However, in some situations it may be regarded as too restrictive limitation. In the general case (eg. when using Gaussian input fuzzy sets) such a requirement is not met and the operation of real numbers division at the output is required, as shown in Eq. (7).

In many publications this issue was analyzed and various solutions have been proposed. For example, the paper [28] proposes the implementation of a division operation based on method of look-up-table (LUT) and addressing with the 6-bit word. Similarly, in the work [44] it was proposed division in which the divisor was rounded to the 8-bit number. As you can easily guess in both cases this resulted in a very low accuracy of the result.

In another work [37] the implementation of this operation on the basis of single precision floating point arithmetic was used. The result is a high accuracy but achieved at the expense of rather low performance. How it was indicated in the cited reference the obtained floating-point divider needs 26 clock cycles

to establish division result. Others floating-point operations like multiplication, addition and subtraction take 5 clock cycles, while similar fixed-point operations takes only one cycle in a typical case. This indicates that the floating-point operations are much less efficient than a fixed-point ones in general. It is also important to note that the floating-point operations consume a lot of hardware resources.

In this paper it is proposed that the division operation required in (7) is performed as in some other works which uses fixed-point arithmetic. In such a case a multiplication by the inverse of the denominator is used instead of the division of two numbers. Determination of the inverse of the denominator is made on the basis of the method of look-up table (LUT). The disadvantage of such a method is that it is necessary to use a large amount of memory to store data in the table with an acceptable accuracy.

In this paper it is proposed to use the simplified 18-bit floating-point numbers to store data in the table. This approach reduces the memory consumption. FPGAs usually have a dedicated Block RAM memory, which are organized as 512 locations of 18-bits words [67]. The proposed simplified floating-point arithmetic is therefore well suited to the optimum utilization of hardware resources.

3 Implementation Results

In our investigation it was considered a problem of hardware implementation of particular parts of a fuzzy structure (FS). Considered structure has four inputs, eight rules and one output. The FS was implemented in the Spartan XC6SLX45-3C FPGA from Xilinx by means of Altium Designer and Xilinx ISE software. To encode the values of the real numbers a 32-bit fixed-point arithmetic were used. Widely known and biggest drawback of fixed-point arithmetic is the limited range and the need for continuous scaling of processed numbers. However, the use of 32-bits width words made it possible to obtain a relatively wide range at the same time fairly good accuracy. Thus, in this case this defect was somewhat minimized. Because of the necessary scaling is closely related to a specific application, this issue will be omitted for the sake of readability of the presentation. It will be presented in detail only in places that are important from the point of view of the presented algorithm.

3.1 Fuzyfication and Inference

According to the method proposed in the previous section operations of fuzzification and inference were carried out in the overall processing of input data. As a result, determination of the output value of the formula (5) requires a series of operations such as multiplication and addition. It is possible to perform these actions both in parallel, series and the in series-parallel mode. Fully parallel implementation of calculations allows us to achieve high performance at the expense of high demand on hardware resources. Serial implementation allows us to reduce the use of hardware resources, but with a significant loss of obtained

processing efficiency. While a semi-serial or a semi-parallel implementation allows for compromise.

In the presented example the semi-serial implementation was used. However, the use of high-performance fixed-point calculations made it possible to achieve high-speed processing.

Elementary operations for the input of the rule (5) are executed in parallel as shown in Fig. 1. The elementary function has three 32-bits width inputs. First two inputs (W1 and W2) are the weights coefficients $w_{i,1}^j$ and $w_{i,2}^j$ respectively, the third input (X) is the input to the FS, i.e. x_i as defined in (5). As a result this function performs several operations in one cycle. The register shown in Fig. 1 acts as a component partial sum according to the formula (5). Initial value of the register is equal to weight w_0^j and it is set in the first clock cycle. Three 32-bit fixed point multipliers (FP_MULTIPLIER_1, 2 and 3) and one 32-bit adder (ADDER_1) generates the output within the second cycle. Using the second 32-bit adder (ADDER_2) and one register the whole squared weighted sum (5) for one rule with four inputs is obtained in the fifth cycle. In the sixth cycle the LUT block indicated as EXP_FUNCTION is used to determine the output value of the nonlinear exponential function. The LUT consists of 1024 words each 12-bits width to store the shape of gausoid function with a reasonable accuracy. Summing up, in the general case the whole process of calculation of rule activation degree requires the following number of clock cycles

$$c_r = 2 + N \tag{8}$$

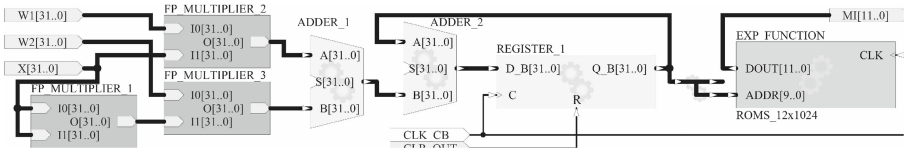

Fig. 1. The hardware implementation of the elementary function.

To calculate the output value y of the FS we need to perform the above described processing for all M rules. This can be done in sequence (serial method) or in parallel (for example with the use of pipelining) to obtain a different speed processing of the implemented system. As mentioned earlier in this paper was carried out the semi-serial implementation method.

3.2 The Defuzification Proces

In the proposed method Fig. 2 shows how all the rules are indicated in order to determine their activation degree μ^j and their consequent $\mu^j \cdot \overline{y}^j$. While, Fig. 3 shows the module used for sequentially processing all rules. Two adders and two registers are used to accumulate values of activation degrees and consequents of

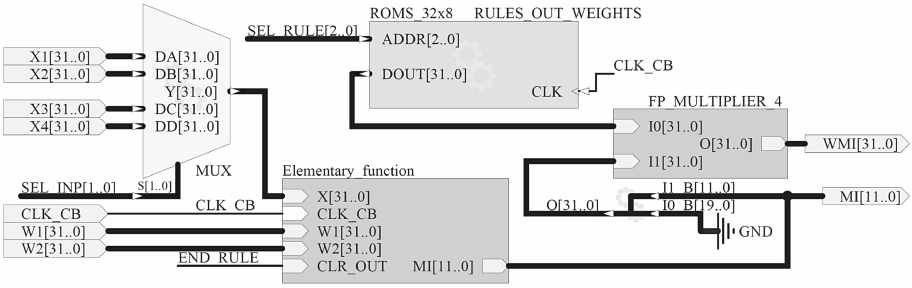

Fig. 2. The hardware implementation of the algorithm used to determine the consequents of fuzzy rules.

all rules. The results, i.e. the nominator and the denominator of the Eq. (7) are obtained within the following number of clock cycles

$$c_{rms} = M \cdot c_r. \tag{9}$$

After the nominator n and denominator d are determined the one extra clock cycle is necessary to calculate the current output value of the FS according to used method of defuzzification (7). First of these two values is multiplied by the reciprocal of the other (Fig. 3) in order to obtain the value of y according to the following formula $y = n \cdot RECIPROCAL\,(d)$.

The RECIPROCAL module used for this purpose has one input and one output which are 12-bits and 27-bits width unsigned words respectively. The input has a fixed point 3.9 bit representation, i.e. three bits for the integer value and nine bits for fraction. This gives the useful range of $d \in (0; 8)$. Since every single rule has the activation degree with a range of $\mu^j \in \langle 0; 1\rangle$ this allows to store the information about the sum of activation degree values for many rules. The upper limit for the used fixed point representation for the reciprocal input is, for example, when eight rules have activation coefficient close to unity which is rather unrealistic in properly designed system.

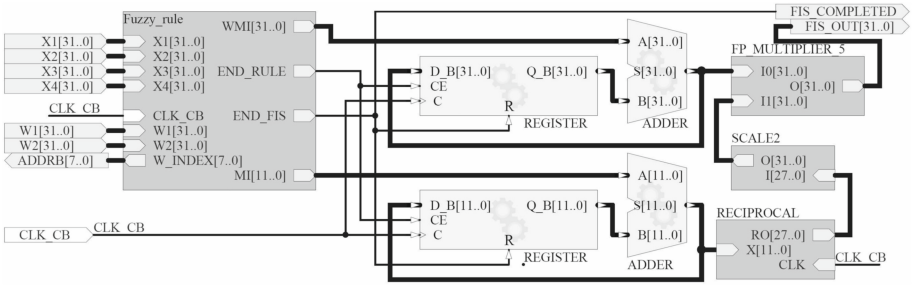

Fig. 3. The block diagram of the implemented fuzzy system with one output.

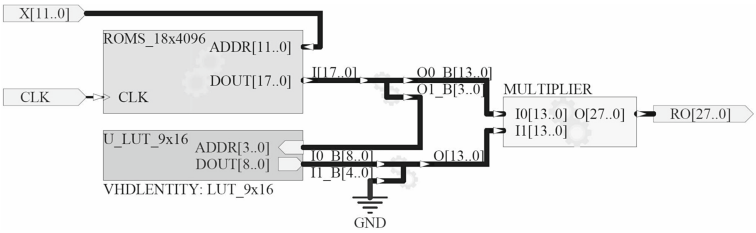

Fig. 4. The method of hardware implementation of the reciprocal function.

The RECIPROCAL module is implemented as an look-up table (LUT) located in a read-only memory and it stories 4096 words (Fig. 4). Value of the input of the module is treated as a 12-bit address, which indexes the table. Each indexed word location stores the result of operation RO = 1/X in a 18-bit simplified floating-point (SFP) format.

The SFP is proposed in this paper a nonstandard format of encoding floating-point numbers. The SFP is an encoding format tailored to a specific application. It allows to reduce the number of bits of a binary word and to simplify their processing. The general idea is derived from the standard IEEE754 but limited to the processing of positive numbers and with a limited range. In the SFP format the 18-bit word is divided into two bitfields: 4-bits for exponent and 14-bits for mantissa. The exponent is a positive number with range of $(0; 15)$. It gives the useful range for floating-point values of $\langle 0; 32768)$ with an acceptable accuracy. For example, the accuracy is about 0.01 % for numbers with a value close to unity or larger. In the paper the exponent is limited to the range of $\langle 0; 8\rangle$ for practical reasons.

The detailed method of processing the SFP numbers is shown in Fig. 4. Two LUTs are used. First (ROMS_18x4096) stores the 18-bit words in SFP format. The second one (U_LUT_9x16) together with the fixed-point multiplier is used to change the SFP format to the fixed-point one. The U_LUT_9x16 is a binary decoder which converts the 4-bits binary number (input) to the 1-of-9 output bits. The fixed-point format which is used on the output of the RECIPROCAL module is compatible with the rest of the system. While the SFP format is used only in the RECIPROCAL module to store the data table. Such approach has allowed to reduce memory consumption by more than 50 %, while maintaining the accuracy and the processing performance at the same level.

3.3 Results

The timing analysis shows that the exemplary FS implemented in the FPGA device is able to work with clock frequency above 50MHz, which gives the reaction time below 1 µs. This allows us to build a FS system, that could be useful for some kind of applications, e.g. hardware emulators.

The implementation results presented in the Table 1 are valid for a system with one output. However, for a system with multiple outputs the resource con-

Table 1. Performance and the FPGA resource usage of the exemplary fuzzy system implemented with the use of the proposed method.

Response time	DSP48A1	Registers	Block RAM	LUTs
49 cycles	21	35	10	627
0.98 µs	(36 %)	(0.1 %)	(9 %)	(2 %)

sumption will be almost the same when the serial implementation is used. Obviously, the response time will be many times larger (proportional to the number of outputs) compared to the system with one output.

4 Summary

In this paper a method of implementation of fuzzy system on FPGA devices was presented. The method applies to a class of fuzzy systems which are functionally equivalent to the radial basis function networks. Thanks to this similarity it was possible to propose the effective methods of such fuzzy systems implementation in FPGA-type programmable systems. For a demonstration of the method the results of the implementation of an exemplary fuzzy system in the FPGA was presented. The results show that the FS system with four inputs, eight rules and one output can work with the processing cycle of less than one microsecond. It makes the proposed solution useful in practice.

Presented solution is highly scalable, because depending on the requirements it is possible to shortening response time at the expense of increase the hardware resources. Similarly, it is possible to increase the number of inputs and outputs and the number of rules of the system.

Acknowledgment. The project was financed by the National Science Centre (Poland) on the basis of the decision number DEC-2012/05/B/ST7/02138.

References

1. Antonio-Mendez, R., de la Cruz-Alejo, J., Peñaloza-Mejia, O.: Fuzzy logic control on FPGA for solar tracking system. In: Ceccarelli, M., Ceccarelli, E.E.H. (eds.) Multibody Mechatronic Systems. Mechanisms and Machine Science, vol. 25, pp. 11–21. Springer, Switzerland (2015)
2. Aghdam, M.H., Heidari, S.: Feature selection using particle swarm optimization in text categorization. J. Artif. Intell. Soft Comput. Res. **5**(4), 231–238 (2015)
3. Aissat, K., Oulamara, A.: A priori approach of real-time ridesharing problem with intermediate meeting locations. J. Artif. Intell. Soft Comput. Res. **4**(4), 287–299 (2014)
4. Akhtar, Z., Rattani, A., Foresti, G.L.: Temporal analysis of adaptive face recognition. J. Artif. Intell. Soft Comput. Res. **4**(4), 243–255 (2014)

5. Bahoura, M., Park, C.-W.: FPGA-implementation of dynamic time delay neural network for power amplifier behavioral modeling. Analog Integr. Circuits Signal Process. **73**, 819–828 (2012)
6. Bartczuk, Ł.: Gene expression programming in correction modelling of nonlinear dynamic objects. ISAT 2015 – Part I. AISC, vol. 429, pp. 125–134. Springer, Switzerland (2016)
7. Bartczuk, Ł., Przybył, A., Koprinkova-Hristova, P.: New method for nonlinear fuzzy correction modelling of dynamic objects. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014, Part I. LNCS, vol. 8467, pp. 169–180. Springer, Heidelberg (2014)
8. Bartczuk, Ł., Rutkowska, D.: Medical diagnosis with type-2 fuzzy decision trees. In: Kącki, E., Rudnicki, M., Stempczyńska, J. (eds.) Computers in Medical Activity. AISC, vol. 65, pp. 11–21. Springer, Heidelberg (2009)
9. Bartczuk, Ł., Rutkowska, D.: Type-2 fuzzy decision trees. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 197–206. Springer, Heidelberg (2008)
10. Bas, E.: The training of multiplicative neuron model based artificial neural networks with differential evolution algorithm for forecasting. J. Artif. Intell. Soft Comput. Res. **6**(1), 5–11 (2016)
11. Bosque, G., del Campo, I., Echanobe, J.: Fuzzy systems, neural networks and neuro-fuzzy systems: a vision on their hardware implementation and platforms over two decades. Eng. Appl. Artif. Intell. **32**, 283–331 (2014)
12. Benzekri, A., Azrar, A.: FPGA-based design process of a fuzzy logic controller for a dual-axis sun tracking system. Arab. J. Sci. Eng. **39**, 6109–6123 (2014)
13. Camargo, E., Aguilar, J.: Advanced supervision of oil wells based on soft computing techniques. J. Artif. Intell. Soft Comput. Res. **4**(3), 215–225 (2014)
14. Chen, M., Ludwig, S.A.: Particle swarm optimization based fuzzy clustering approach to identify optimal number of clusters. J. Artif. Intell. Soft Comput. Res. **4**(1), 43–56 (2014)
15. Cheng, S., Shi, Y., Qin, Q., Zhang, Q., Bai, R.: Population diversity maintenance in brain storm optimization algorithm. J. Artif. Intell. Soft Comput. Res. **4**(2), 83–97 (2014)
16. Cierniak, R., Rutkowski, L.: On image compression by competitive neural networks and optimal linear predictors. Signal Process. Image Commun. **156**, 559–565 (2000)
17. Cpałka, K., Łapa, K., Przybył, A.: A new approach to design of control systems using genetic programming. Inf. Technol. Control **44**(4), 433–442 (2015)
18. Cpałka, K., Rutkowski, L.: Flexible Takagi-Sugeno fuzzy systems. In: Proceedings of the International Joint Conference on Neural Networks 2005, Montreal, pp. 1764–1769 (2005)
19. Cpałka, K., Rutkowski, L.: Flexible Takagi-Sugeno neuro-fuzzy structures for nonlinear approximation. WSEAS Trans. Syst. **4**(9), 1450–1458 (2005)
20. Cpałka, K., Rutkowski, L.: A new method for designing and reduction of neuro-fuzzy systems. In: Proceedings of the 2006 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence, WCCI 2006), Vancouver, pp. 8510–8516 (2006)
21. Cpalka, K.: A method for designing flexible neuro-fuzzy systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 212–219. Springer, Heidelberg (2006)
22. Cpałka, K.: On evolutionary designing and learning of flexible neuro-fuzzy structures for nonlinear classification. Nonlinear Anal. Series A Theory Methods Appl. **71**, 1659–1672 (2009). Elsevier

23. Cpałka, K., Rebrova, O., Nowicki, R., Rutkowski, L.: On design of flexible neuro-fuzzy systems for nonlinear modelling. Int. J. Gen. Syst. **42**(6), 706–720 (2013)
24. Cpałka, K., Zalasiński, M.: On-line signature verification using vertical signature partitioning. Expert Syst. Appl. **41**(9), 4170–4180 (2014)
25. Cpałka, K., Zalasiński, M., Rutkowski, L.: New method for the on-line signature verification based on horizontal partitioning. Pattern Recognit. **47**, 2652–2661 (2014)
26. Cpałka, K., Łapa, K., Przybył, A., Zalasiń ski, M.: A new method for designing neuro-fuzzy systems for nonlinear modelling with interpretability aspects. Neurocomputing **135**, 203–217 (2014)
27. Cpałka, K., Zalasiński, M., Rutkowski, L.: A new algorithm for identity verification based on the analysis of a handwritten dynamic signature. Appl. Soft Comput. **43**, 47–56 (2016). http://dx.doi.org/10.1016/j.asoc.2016.02.017
28. Deliparaschos, K.M., Nenedakis, F.I., Tzafestas, S.G.: Design and implementation of a fast digital fuzzy logic controller using FPGA technology. J. Intell. Robot. Syst. **45**, 77–96 (2006)
29. Dziwiński, P., Bartczuk, Ł., Przybył, A., Avedyan, E.D.: A new algorithm for identification of significant operating points using swarm intelligence. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014, Part II. LNCS, vol. 8468, pp. 349–362. Springer, Heidelberg (2014)
30. Dziwiński, P., Avedyan, E.D.: A new approach to nonlinear modeling based on significant operating points detection. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. LNCS, vol. 9120, pp. 364–378. Springer, Heidelberg (2015)
31. Gałkowski, T., Rutkowski, L.: Nonparametric recovery of multivariate functions with applications to system identification. Proc. IEEE **73**(5), 942–943 (1985)
32. Gałkowski, T., Rutkowski, L.: Nonparametric fitting of multivariate functions. IEEE Trans. Autom. Control **31**(8), 785–787 (1986)
33. Gdaim, S., Mtibaa, A., Mimouni, M.F.: Design and experimental implementation of DTC of an induction machine based on fuzzy logic control on FPGA. IEEE Trans. Fuzzy Syst. **23**(3), 644–655 (2015)
34. Gręblicki, W., Rutkowski, L.: Density-free Bayes risk consistency of nonparametric pattern recognition procedures. Proc. IEEE **69**(4), 482–483 (1981)
35. Li, H., Gupta, M.: Fuzzy Logic and Intelligent Systems, pp. 50–55. Kluwer Academic Publishers, Boston (1995)
36. Jang, J.S.R., Sun, C.T.: Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Trans. Neural Netw. **4**(1), 156–159 (1993)
37. Kluska, J., Hajduk, Z.: Hardware implementation of P1-TS fuzzy rule-based systems on FPGA. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part I. LNCS, vol. 7894, pp. 282–293. Springer, Heidelberg (2013)
38. Korytkowski, M., Rutkowski, L., Scherer, R.: From ensemble of fuzzy classifiers to single fuzzy rule base classifier. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 265–272. Springer, Heidelberg (2008)
39. Korytkowski, M., Rutkowski, L., Scherer, R.: Fast image classification by boosting fuzzy classifiers. Inf. Sci. **327**, 175–182 (2016)
40. Li, X., Er, M.J., Lim, B.S., Zhou, J.H., Gan, O.P., Rutkowski, L.: Fuzzy regression modeling for tool performance prediction and degradation detection. Int. J. Neural Syst. **20**(05), 405–419 (2010)

41. Łapa, K., Cpałka, K., Wang, L.: New method for design of fuzzy systems for nonlinear modelling using different criteria of interpretability. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014, Part I. LNCS, vol. 8467, pp. 217–232. Springer, Heidelberg (2014)
42. Łapa, K., Przybył, A., Cpałka, K.: A new approach to designing interpretable models of dynamic systems. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part II. LNCS, vol. 7895, pp. 523–534. Springer, Heidelberg (2013)
43. Łapa, K., Zalasiński, M., Cpałka, K.: A new method for designing and complexity reduction of neuro-fuzzy systems for nonlinear modelling. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part I. LNCS, vol. 7894, pp. 329–344. Springer, Heidelberg (2013)
44. Hassan, M.Y., Sharif, W.F.: Design of FPGA based PID- like fuzzy controller for industrial applications. IAENG Int. J. Comput. Sci. **34**(2), 1–7 (2007)
45. Osowski, S.: Sieci neuronowe w ujeciu algorytmicznym, pp. 160–188. WNT, Warszawa (1996)
46. Poon, J., Haessig, P., Hwang, J., Celanovic, I.: High-speed hardware-in-the loop platform for rapid prototyping of power electronics systems. In: 2010 IEEE Conference on Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES), pp. 420–424 (2010)
47. Poplawski, M., Bialko, M.: Implementation of fuzzy logic controller in FPGA circuit for guiding electric wheelchair. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS, vol. 7268, pp. 216–222. Springer, Heidelberg (2012)
48. Przybył, A., Cpałka, K.: A new method to construct of interpretable models of dynamic systems. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS, vol. 7268, pp. 697–705. Springer, Heidelberg (2012)
49. Rutkowski, L.: Sequential estimates of probability densities by orthogonal series and their application in pattern classification. IEEE Trans. Syst. Man Cybern. **10**(12), 918–920 (1980)
50. Rutkowski, L.: On nonparametric identification with prediction of time-varying systems. IEEE Trans. Autom. Control **29**(1), 58–60 (1984)
51. Rutkowski, L.: Nonparametric identification of quasi-stationary systems. Syst. Control Lett. **6**(1), 33–35 (1985)
52. Rutkowski, L.: Real-time identification of time-varying systems by non-parametric algorithms based on Parzen kernels. Int. J. Syst. Sci. **16**(9), 1123–1130 (1985)
53. Rutkowski, L.: A general approach for nonparametric fitting of functions and their derivatives with applications to linear circuits identification. IEEE Trans. Circuits Syst. **33**(8), 812–818 (1986)
54. Rutkowski, L.: Application of multiple Fourier-series to identification of multivariable non-stationary systems. Int. J. Syst. Sci. **20**(10), 1993–2002 (1989)
55. Rutkowski, L.: Adaptive probabilistic neural networks for pattern classification in time-varying environment. IEEE Trans. Neural Netw. **15**(4), 811–827 (2004)
56. Rutkowski, L., Cpałka, K.: Flexible structures of neuro-fuzzy systems, Quo Vadis Computational Intelligence. Fuzziness and Soft Computing, vol. 54, pp. 479–484. Springer (2000)
57. Rutkowski, L., Cpałka, K.: Compromise approach to neuro-fuzzy systems. In: Sincak, P., Vascak, J., Kvasnicka, V., Pospichal, J. (eds.) Intelligent Technologies - Theory and Applications, vol. 76, pp. 85–90. IOS Press, Amsterdam (2002)

58. Rutkowski, L., Cpałka, K.: A neuro-fuzzy controller with a compromise fuzzy reasoning. Control and Cybern. **31**(2), 297–308 (2002)
59. Rutkowski, L., Przybył, A., Cpałka, K., Er, M.J.: Online speed profile generation for industrial machine tool based on neuro-fuzzy approach. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part II. LNCS, vol. 6114, pp. 645–650. Springer, Heidelberg (2010)
60. Rutkowski, L., Pietruczuk, L., Duda, P., Jaworski, M.: Decision Trees for mining data streams based on the McDiarmid's bound. IEEE Trans. Knowl. Data Eng. **25**(6), 1272–1279 (2013)
61. Rutkowski, L., Jaworski, M., Pietruczuk, L., Duda, P.: Decision trees for mining data streams based on the Gaussian approximation. IEEE Trans. Knowl. Data Eng. **26**(1), 108–119 (2014)
62. Rutkowski, L., Jaworski, M., Pietruczuk, L., Duda, P.: The CART decision tree for mining data streams. Inf. Sci. **266**, 1–15 (2014)
63. Rutkowski, L., Rafajłowicz, E.: On optimal global rate of convergence of some nonparametric identification procedures. IEEE Trans. Autom. Control **34**(10), 1089–1091 (1989)
64. Starczewski, J.T., Bartczuk, Ł., Dziwiński, P., Marvuglia, A.: Learning methods for type-2 FLS based on FCM. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part I. LNCS, vol. 6113, pp. 224–231. Springer, Heidelberg (2010)
65. Tomera, M.: Porównanie jakości pracy trzech algorytmów typu PID: liniowego, rozmytego i neuronowego. Automatyka, Elektryka, Zakłócenia **6**, 59–77 (2011). (in Polish)
66. Wang, L., Mendel, J.M.: Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans. Neural Netw. **3**(5), 807–814 (1992)
67. Xilinx Spartan-6 FPGA User Guides. UG389 v1.5, UG389 v1.2 (2014). http://www.xilinx.com/support/documentation/user_guides/
68. Zalasiński, M., Cpałka, K.: Novel algorithm for the on-line signature verification. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS, vol. 7268, pp. 362–367. Springer, Heidelberg (2012)
69. Zalasiński, M., Cpałka, K.: Novel algorithm for the on-line signature verification using selected discretization points groups. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part I. LNCS, vol. 7894, pp. 493–502. Springer, Heidelberg (2013)
70. Zalasiński, M., Łapa, K., Cpałka, K.: New algorithm for evolutionary selection of the dynamic signature global features. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part II. LNCS, vol. 7895, pp. 113–121. Springer, Heidelberg (2013)
71. Zalasiński, M., Cpałka, K., Er, M.J.: New method for dynamic signature verification using hybrid partitioning. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014, Part II. LNCS, vol. 8468, pp. 216–230. Springer, Heidelberg (2014)
72. Zalasiński, M., Cpałka, K., Hayashi, Y.: New method for dynamic signature verification based on global features. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014, Part II. LNCS, vol. 8468, pp. 231–245. Springer, Heidelberg (2014)

73. Zalasiński, M., Cpałka, K., Hayashi, Y.: New fast algorithm for the dynamic signature verification using global features values. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. LNCS, vol. 9120, pp. 175–188. Springer, Heidelberg (2015)
74. Zalasiński, M., Cpałka, K., Er, M.J.: A new method for the dynamic signature verification based on the stable partitions of the signature. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. LNCS, vol. 9120, pp. 161–174. Springer, Heidelberg (2015)