Advances in Computational Mathematics (2019) 45:1807–1823
https://doi.org/10.1007/s10444-018-09659-5

Parallel, asynchronous, fuzzy logic systems realized in CMOS technology

Tomasz Talaśka[1]

Received: 30 September 2018 / Accepted: 18 December 2018 /
Published online: 4 February 2019

Abstract

Fuzzy systems play an important role in many industrial applications. Depending on the application, they can be implemented using different techniques and technologies. Software implementations are the most popular, which results from the ease of such implementations. This approach facilitates modifications and testing. On the other hand, such realizations are usually not convenient when high data rate, low cost per unit, and large miniaturization are required. For this reason, we propose efficient, fully digital, parallel, and asynchronous (clock-less) fuzzy logic (FL) systems suitable for the implementation as ultra low-power-specific integrated circuits (ASICs). On the basis of our former work, in which single FL operators were proposed, here we demonstrate how to build larger structures, composed of many operators of this type. As an example, we consider Lukasiewicz neural networks (LNN) that are fully composed of selected FL operators. In this work, we propose FL OR, and AND Lukasiewicz neurons, which are based on bounded sum and bounded product FL operators. In the comparison with former analog implementations of such LNNs, digital realization, presented in this work, offers important advantages. The neurons have been designed in the CMOS 130nm technology and thoroughly verified by means of the corner analysis in the HSpice environment. The only observed influence of particular combinations on the process, voltage, and temperature parameters was on delays and power dissipation, while from the logical point of view, the system always worked properly. This shows that even larger FL systems may be implemented in this way.

Keywords Fuzzy logic systems · FL operators · FL neural networks · Asynchronous circuits · Parallel circuits · CMOS implementation

Communicated by: Pavel Solin

✉ Tomasz Talaśka
tomasz.talaska@gmail.com

[1] Faculty of Telecommunication, Computer Science and Electrical Engineering,
UTP University of Science and Technology, ul. Kaliskiego 7, 85-796, Bydgoszcz, Poland

Mathematics Subject Classification (2010) 68T27

1 Introduction

The popularity of fuzzy logic systems (FLS) results from the fact that the real world is analog (multivalued). Additionally, in solving real problems, we use a fuzzy approach in a natural way. Bivalent logic is often insufficient, as its hard selection between the TRUE and the FALSE values may lead to curious situations. For example, a group of children may be divided into low and tall on the basis of their height. If we define the limits between two sets, for example, as 110 cm, then in bivalent logic children with the heights of 109.99 cm and 110.01 cm will belong to two different sets. Fuzzy sets that allow the two children to belong to both sets with different percentage are more natural in such situations.

Thanks to various interesting properties of the FLSs, they have found wide application in various areas, for example, in control systems [1, 2], in electrical engineering [3, 4], in planning and prognostic [5], and in aviation [6]. FLSs find application in medical health care systems, used for example in human motion capture [7], in which they are counterparts to systems that are based on conventional positioning algorithms [8]. In automotive area, the use of the FL is investigated in the context of a support for vehicle-to-infrastructure (V2I) communication [9, 10]. In such applications, systems of this type may help in vehicle positioning on the roads [9] that is essential from the point of view of autonomous driving safety, especially in dense urban environment [11]. In the last two described areas, i.e., in human motion capture systems and in recently developed automotive applications, one of paramount features is reduced energy consumption of supporting and controlling systems. For this reason, the results presented in this work match the described applications.

FLSs may cooperate with other neural networks (NNs) [12]. In neuro-fuzzy networks [13, 14], for example, the NN supports the work of the fuzzy system. An interesting feature of these networks is their ability to use fuzzy inference methods to determine the values of the output signals. For example, NNs may be used for an adaptive selection of some parameters of the fuzzy systems, such as shapes and locations of the membership functions (MFs).

The works in this area are carried out at different levels that include mathematical investigations from one hand, and the ways of the implementation, on the other hand. In the latter group of works, the most important are efficient hardware implementations. In this work, we focus on the second aspect. In particular, we focus on the transistor level implementation of such systems that offer the highest flexibility. Fuzzy systems are commonly realized in software, which is due to the ease and flexibility of such implementation. However, many industry applications require miniaturization, low energy consumption, and low cost per unit. In such cases, hardware implementations may be a better option [15]. This approach includes programmable devices such as microcontrollers (μC) [16] and Field Programmable Gate Array (FPGA) devices [15, 17, 18]. They are also reported full custom transistor level systems of this type (ASIC), analog [19–21], digital [22], or mixed [23]. In

case of the analog approach, the circuits are usually realized using the current-mode technique, as in this case the summing and subtracting operations (commonly used in FL) may be realized simply in junctions.

In the comparison with analog approach, digital realization also offers some important advantages. Circuits of this type feature high noise immunity and low sensitivity to the variation of transistor parameters. Furthermore, digital data can be easily stored even for a long period of time that is different than in typical analog solutions. This allows for accurate and reliable data and signal processing and facilitates the realization of even very large, programmable, multi-stage fuzzy data processing.

The paper is organized as follows. In the next section, we present basics of selected FL operators, as well as the proposed hardware implementation of these operators. Then, we propose also larger fuzzy logic systems. We focus, in particular, on Lukasiewicz neural networks as an example of the systems that are fully based on the FL operators. In the following section, we present transistor level verification of the realized circuits. Finally, the conclusions are formulated in the last section.

2 Fuzzy logic network (FLN) and their implementation in hardware

In the literature, one can find a dozen different fuzzy logic operators (FLO) [20, 21, 24, 25]. As described earlier, FLOs may be realized using various techniques: analog, digital and mixed. Each of them techniques offers some advantages and disadvantages. For example, in the analog approach, especially in the current mode, it is easy to carry out the addition and subtraction operation. On the other hand, in this case, the circuits have to be realized very carefully to avoid or minimize an impact of various phenomena typical for analog realizations—transistor mismatch, leakage, charge injection, etc. In digital technology, however, data processing may be much faster especially in the most advanced recent technologies. Circuits of this type exhibit a better noise immunity. Usually, there is also no problem with data storage over long periods of time.

In our previous work [24, 25], we proposed an efficient transistor level implementation of all FOs known in the literature in the form of digital parallel and asynchronous circuits. The operators are composed of single logic gates and more complex blocks, such as multi-bit adders, multi-bit subtractors that may serve as a digital comparator as well. The proposed circuits can be easily scaled to differed technologies. The proposed FLOs became a basis for the works presented in this paper that aimed at the realization of larger FLSs. Systems of this type may be used in different applications, and thus may be composed of different basic operators. As an example, in this work, we focus on fuzzy logic Lukasiewicz neural networks that are fully composed of such operators. As it is presented in the paper, such systems may also be implemented as fully parallel and asynchronous solutions, that is an important advantage here.

There are two types of the LNN, namely the AND-OR and the OR-AND ones. They are build of the AND and OR FL neurons. These neurons, in turn, are based on such FLOs as the bounded sum (operator *or*) and bounded product (operator *and*), described below.

2.1 Selected fuzzy logic operators

A transistor level realization of a full set of the FL operators have been described in our previous works. For this reason, here we present only selected operators, those that have been directly applied in the logical Lukasiewicz neurons, presented latter in this work. The first of them is the Bounded Product (BP) operator, which may be described in the following way:

$$Y_{A\odot B} = \max[0, (A + B - \mathbb{1})] \tag{1}$$

The basic operations that are performed in this case are the determination the maximum value, the addition and the subtraction. It is important to clarify the meaning of the $\mathbb{1}$ factor. This is a normalized value that depends on the type of the implementation and the signal resolution. It means the maximum possible signal value. For example, in case of the current mode analog implementation, it would be the maximum allowed value of each of the input signals (current). In case of digital implementations, the value of this signal depends on the signal resolution. For example, cases of 8 and 16 bits, the $\mathbb{1}$ factor will be 255 and 65535, respectively.

This BP operator works as follows. If the sum of its inputs is less than $\mathbb{1}$, then its output is 0. In all other cases, the output is the sum of its inputs minus $\mathbb{1}$. When both inputs have maximum values, the output signal reaches $\mathbb{1}$.

The second operator in the LNN is the Bounded Sum (BS), which is expressed as follows:

$$Y_{A\oplus B} = \min[\mathbb{1}, A + B] \tag{2}$$

The operations that are used in this case are the minimum function signal and the summing operation. In this case, if the sum of the operator inputs is less than $\mathbb{1}$, the value of this sum is provided to the output. When the sum become larger than $\mathbb{1}$, then the output saturates at $\mathbb{1}$. Figure 1 shows our proposed structures of both described operators.

The BP operator is based on two multi-bit full adders (MBFAs), composed of a chain of one bit full adders (1BFAs). Since 1BFA is one of basic building blocks in the proposed FL operators, in Fig. 2, we present selected transistor level implementations of such a block. Figure 2a presents a conventional approach, in which the circuit is composed of distinct binary logic gates, as visible in Fig. 3. In this case, the adder is built of 38 transistors. In Fig. 2b, we present a simpler circuit of this type, composed of 30 transistors, which has been used in our realization. In the literature one can also find 1BFAa composed of even smaller numbers of transistors. However, we observed some problems when many MBFAs composed of such 1BFA were connected in a chain. Our assumption, however, is to create larger fuzzy systems, composed of many FL operators. For this reason, we decided to use a more stable solution, even at the expense of a bit larger number of transistors.

The circuit shown in Fig. 2b, as well as the MBFAs built of such circuits, has been used by us in a prototype chip, designed by us in the CMOS 130 nm technology for other purposes. The chip has been verified by means of laboratory measurements. Since the realized MBFA passed the tests, we used it also in the current work.

First of the described operations, $A+B$, is performed using the MBFA. The second operation (the subtraction of the $\mathbb{1}$ factor) is also based on the summing circuit. We

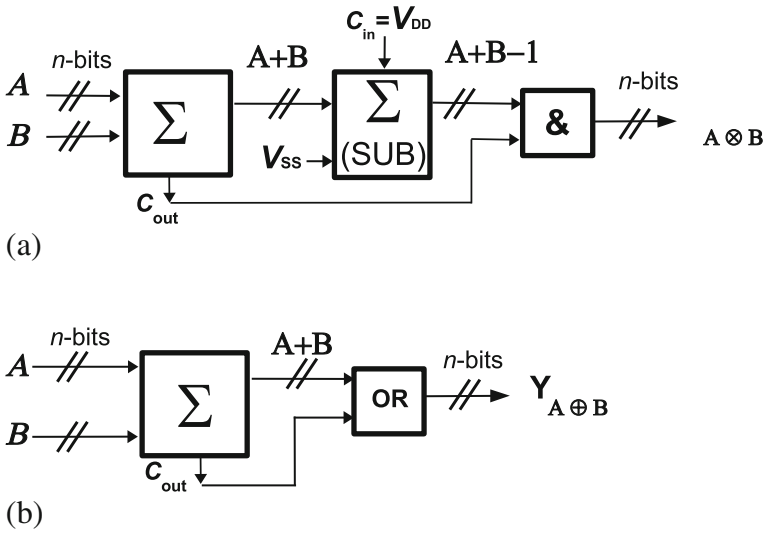

Fig. 1 Proposed realization of the: **a** FL bounded product operator, and **b** FL bounded sum operator

reverse all bits in the $\mathbb{1}$, add the resultant signal to $A + B$, and complement it by addition of logical '1'. The advantage of this approach is that independently on the signal resolution, the reversed and complemented value of $\mathbb{1}$ always equals '1', so it is sufficient to add the '1' signal at the least significant position.

If the C_{out} (carry out) signal from the first MBFA equals '0', which means that the sum of the A and B signals is $\leq \mathbb{1}$, the final result of the $A + B - \mathbb{1}$ operation is also ≤ 0. In this case, the output signal of the overall circuit equals '0'.

In the BP operator, the MBFA is used to calculate the sum of both input signals. If $A + B > \mathbb{1}$, then the C_{OUT} signal becomes '1' and this signal through the OR (bivalent) logic gate sets all output signals to '1'. The resolution of the output signal equals the resolution of the A and the B signals. This allows to avoid the situation, in which the operator provides to the output signal with greater resolution than one of its inputs.

Gate level diagrams of both proposed and used operators are shown in Fig. 3. The circuits are presented for the situation, in which conventional 1BFA are used, shown in Fig. 2a.

In case of the LNN, the Bounded Product operator is called the *and* operation, whereas the Bounded Sum operator is referred to as the *or* operation. In the further part of this work these operators will be denoted in this way, for a better clarity.

In the literature, various realizations of the full adder can be found. The differences are mainly visible in the number of transistors used, and these in turn to some extent result from different implementation of the XOR gate. However, the adder itself is not the main topic of this work, which means that in the FL operators presented in this work, different adders may be used, without an impact on functional behavior of the proposed FL operators.

In the context of fuzzy logic systems, and state-of-the-art study, it is worth mentioning multivalued adders reported in the literature. Such adders are interesting

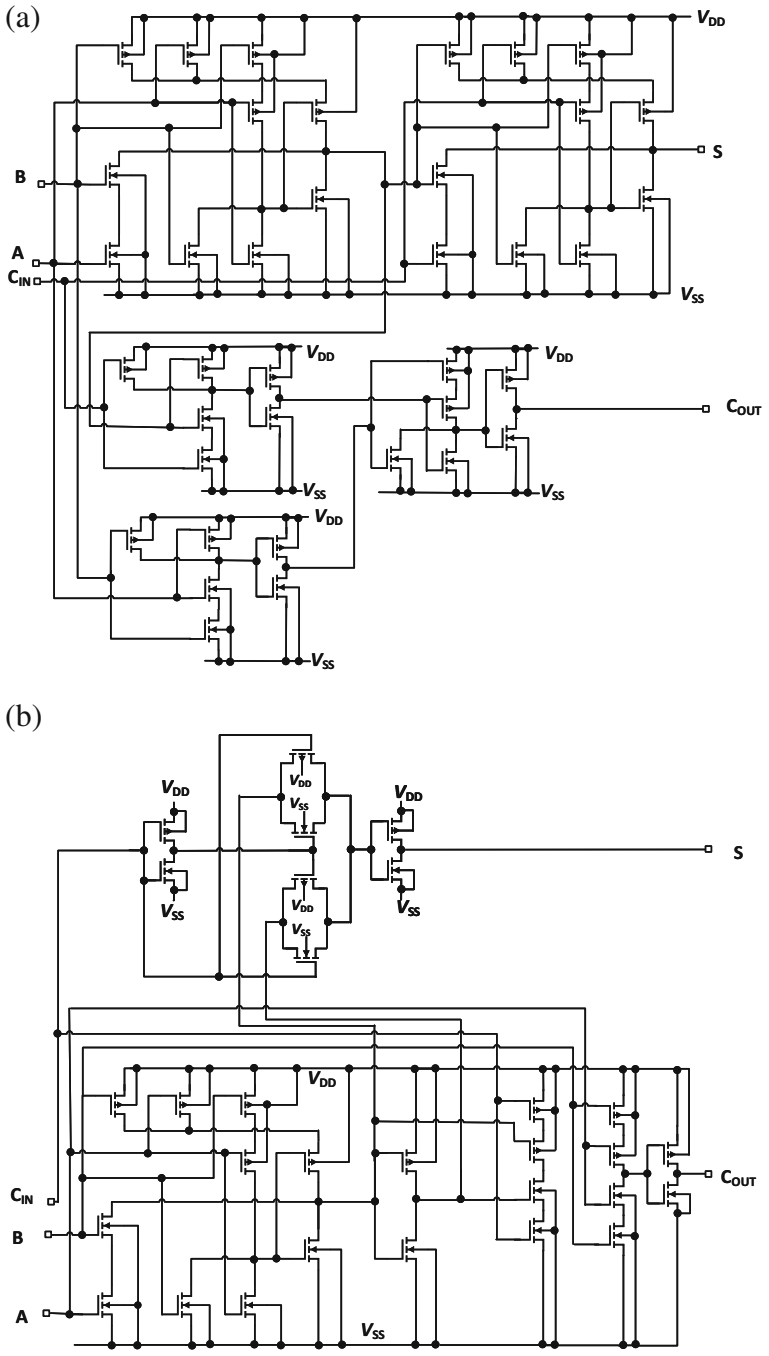

Fig. 2 Selected transistor level realizations of the 1-bit full adder: **a** a conventional approach in which the circuit is composed of 38 transistors, **b** the circuit composed of 30 transistors, used in presented realization

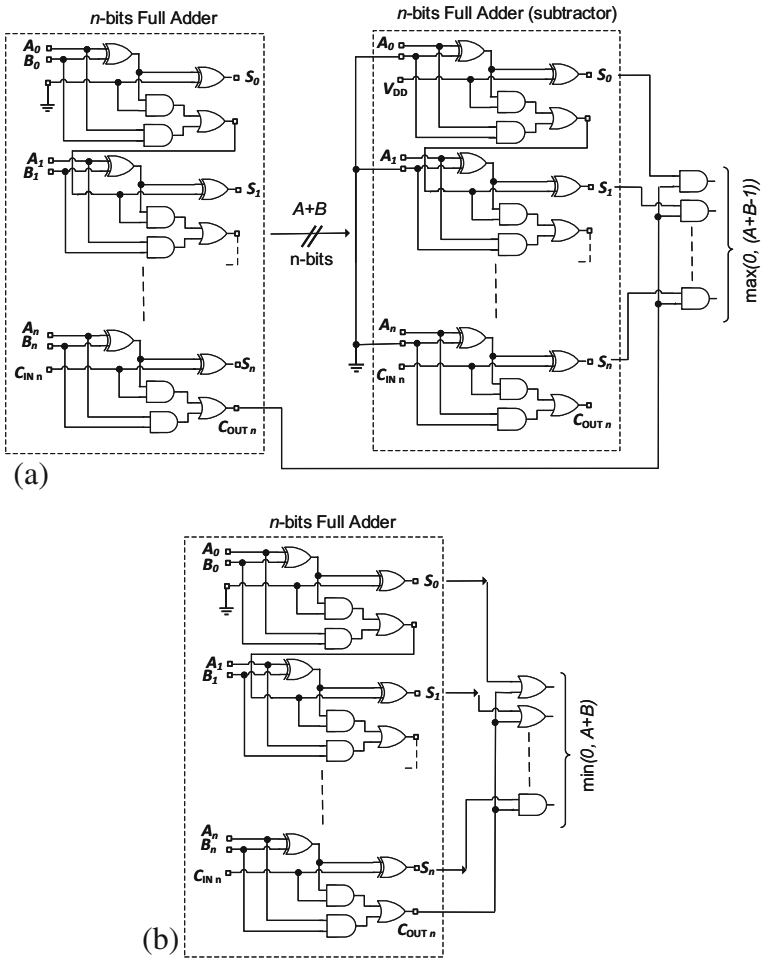

Fig. 3 Gate level implementation of both proposed operators: **a** the BP operator, and **b** the BS operator. For a better illustration, we present these operators as if they were built of conventional 1BFA, shown in Fig. 2a

group of circuits, which allow to implement FL systems in different ways. Example circuits of this, reported in [26], are a multivalued (fuzzy) analog adders operating in the voltage and the current mode. These adders, unlike the classic solutions used for adding voltages or currents, in addition to the S (SUM) signal, also have the C_{OUT} (Carry Out) output terminal, while both the S and the C_{OUT} signals are multivalued. These solutions, interesting itself, may also be used for digital implementation under certain conditions. The use of fuzzy adders, as some components of a large fuzzy system, requires that other components cooperating with such blocks should be able to handle multivalued signals.

In the literature, one can also find various realizations of the full Bounded Product and the Bounded Sum operators. In [27], an interesting implementation of such operators is reported. In [27] work, the output block is built based on a multi-bit

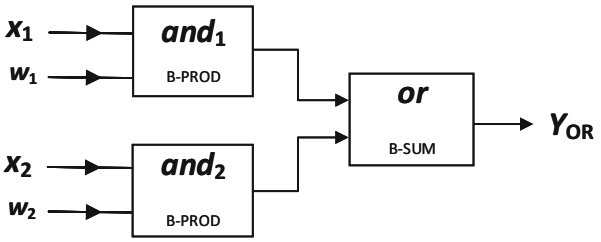

Fig. 4 Lukasiewicz fuzzy logic OR neuron

multiplexer. Compared to the operators presented in [27], this paper proposes solutions, in which the multiplexers are replaced with classic logic gates.

2.2 Proposed realization of the FL OR and AND neurons

The OR neuron performs the *and* logic aggregation of its input signals, denoted as

$$x = [x_1, x_2, x_3 ... x_n]^T \tag{3}$$

with their corresponding connection (weights), denoted as

$$w = [w_1, w_2, w_3 ... w_n].^T \tag{4}$$

The partial results of particular *and* operations are then summed using the *or* operation (hence the name of the neuron) [21].

The AND neuron is the counterpart of the OR neuron and is created by reversing all basic FL operations. First, it performs the *or* FL aggregation of its inputs (Eq. 3 with their corresponding weights (Eq. 4). The partial results of these operations are summed using the *and* operation (hence the name of the neuron). Block diagrams of the OR and the AND neurons are shown in Figs. 4 and 5, respectively.

The output of the Lukasiewicz OR neuron may be described, as follows:

$$Y_{\mathrm{OR}} = (x_1 \, and \, w_1) \, or \, (x_2 \, and \, w_2) \, or \, (x_3 \, and \, w_3) ..... \, or \, (x_n \, and \, w_n) \tag{5}$$

while the Lukasiewicz AND neuron as below:

$$Y_{\mathrm{AND}} = (x_1 \, or \, w_1) \, and \, (x_2 \, or \, w_2) \, and \, (x_3 \, or \, w_3) ..... \, and \, (x_n \, or \, w_n) \tag{6}$$

An explanation is required how the output *or* and *and* operations in both neurons are realized in the case of a larger number of the input signals. In this case, the use

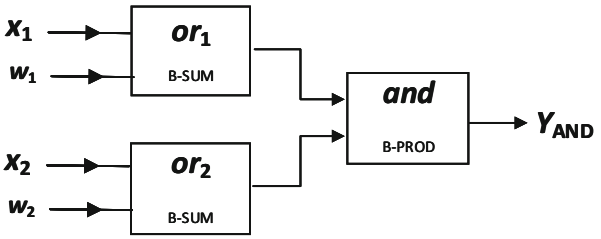

Fig. 5 Lukasiewicz fuzzy logic AND neuron

of only a single operator is not sufficient. In the proposed implementation, these operators are implemented as binary trees consisting of the basic *or* and the *and* operators. The number of such operators in the binary tree always equals the number of the inputs minus one. The number of layers in the tree increases moderately with the number of the inputs and is expressed as $\log_2 M$, where M is the number of the input signals. As a result, the drop in the speed of the neurons is relatively small with increasing the length of the input vector.

It is worth to explain the influence of the weights on the outputs of the neurons. Let us consider extreme cases of the values of the weights. In case of the OR neuron, when all its weights equal 0 ($w_1 = 0, w_2 = 0, \ldots, w_n = 0$), the output signal of the neuron is insensitive to particular input x signals, independently on their values. In this case, the output of the OR neuron, $Y_{\mathrm{OR}} = 0$, in each case. On the other hand, when all the neuron weights equal $\mathbb{1}$ ($w_1 = \mathbb{1}, w_2 = \mathbb{1}, \ldots, w_n = \mathbb{1}$) the output of the OR neuron is expressed as: $Y_{OR} = x_1 \, or \, x_2 \, or \ldots$.

Another situation appears when we consider the AND neuron. In this case, when the values of all weights are $\mathbb{1}$ ($w_1 = \mathbb{1}, w_2 = \mathbb{1}, \ldots, w_n = \mathbb{1}$), the output signal from this neuron, $Y_{\mathrm{AND}} = \mathbb{1}$, independently on the values of the input x signals. In the opposite case, when all neuron weights are 0, the output signal of the neuron is expressed as $Y_{\mathrm{AND}} = x_1 \, and \, x_2 \, and \ldots$.

Taking into account formulas 1 and 2, the output of the OR neuron, shown in Fig. 4, may be expressed as follows:

$$Y_{\mathrm{OR}} = \min[\mathbb{1}, \max(0, x_1 + w_1 - \mathbb{1}) + \max(0, x_2 + w_2 - \mathbb{1})] \tag{7}$$

Analogously, taking into account formulas 1 and 2, the output of the AND neuron, shown in Fig. 5, may be expressed as

$$Y_{\mathrm{AND}} = \max[0, \min(\mathbb{1}, x_1 + w_1) + \min(\mathbb{1}, x_2 + w_2) - \mathbb{1}] \tag{8}$$

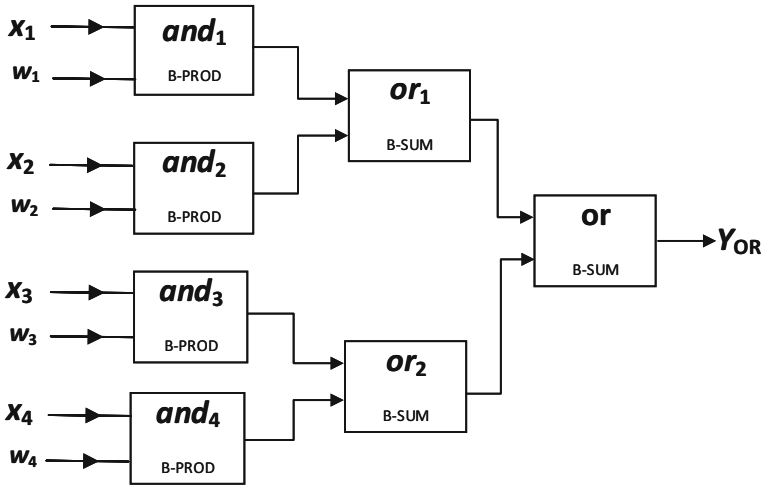

Fig. 6 An example FL OR neuron with four inputs implemented with the use of the parallel and asynchronous binary tree

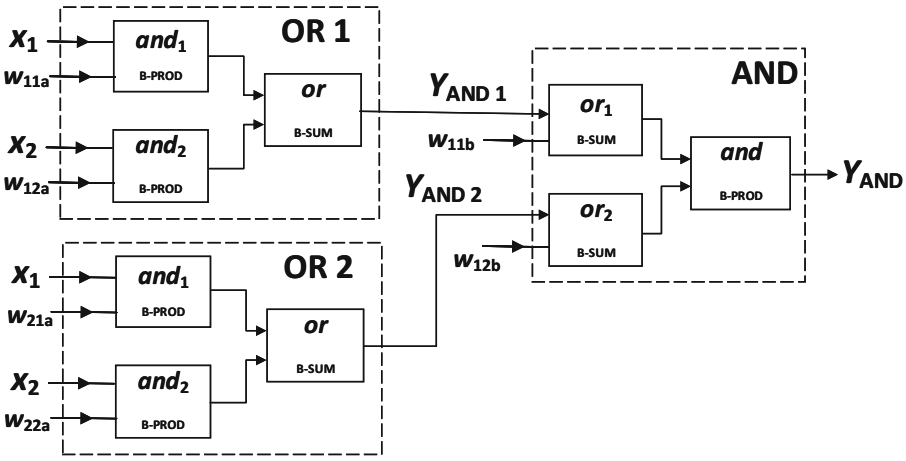

Fig. 7 The proposed two-layer, Lukasiewicz OR-AND neural network

The considerations discussed above concern neurons that have two inputs, i.e., x_1 and x_2. In case of the software implementation, the output operations are realized in an iterative fashion. In the proposed parallel implementation, it may be performed asynchronously, using the binary tree described earlier. Such an implementation is shown in Fig. 6. For an example case of four inputs, the output signals for the OR and the AND neurons may be expressed, respectively, as follows:

$$\begin{aligned} Y_{\mathrm{OR}} = & \min\left[\mathbb{1}, \min\left(\mathbb{1}, \max\left(0, x_1 + w_1 - \mathbb{1}\right) + \max\left(0, x_2 + w_2 - \mathbb{1}\right)\right)\right. \\ & \left. + \min\left(\mathbb{1}, \max(0, x_3 + w_3 - \mathbb{1}) + \max\left(0, x_4 + w_4 - \mathbb{1}\right)\right)\right] \end{aligned} \tag{9}$$

and:

$$\begin{aligned} Y_{\mathrm{AND}} = & \max\left[0, \max\left(0, \min\left(\mathbb{1}, x_1 + w_1\right) + \min\left(\mathbb{1}, x_2 + w_2\right) - \mathbb{1}\right)\right. \\ & \left. + \max\left(0, \min(\mathbb{1}, x_3 + w_3) + \min\left(\mathbb{1}, x_4 + w_4\right) - \mathbb{1}\right) - \mathbb{1}\right] \end{aligned} \tag{10}$$

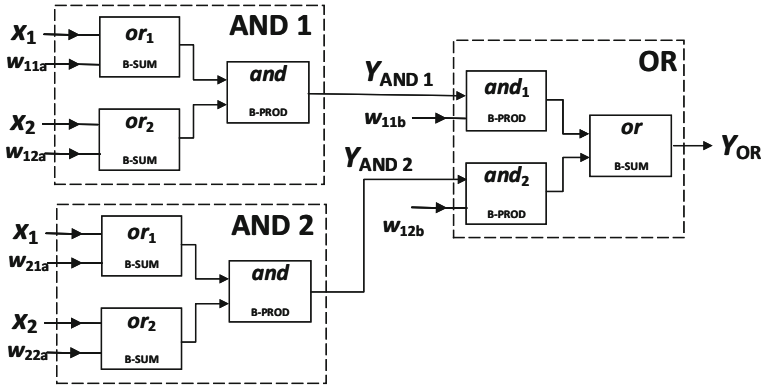

Fig. 8 The proposed two-layer, Lukasiewicz AND-OR neural network

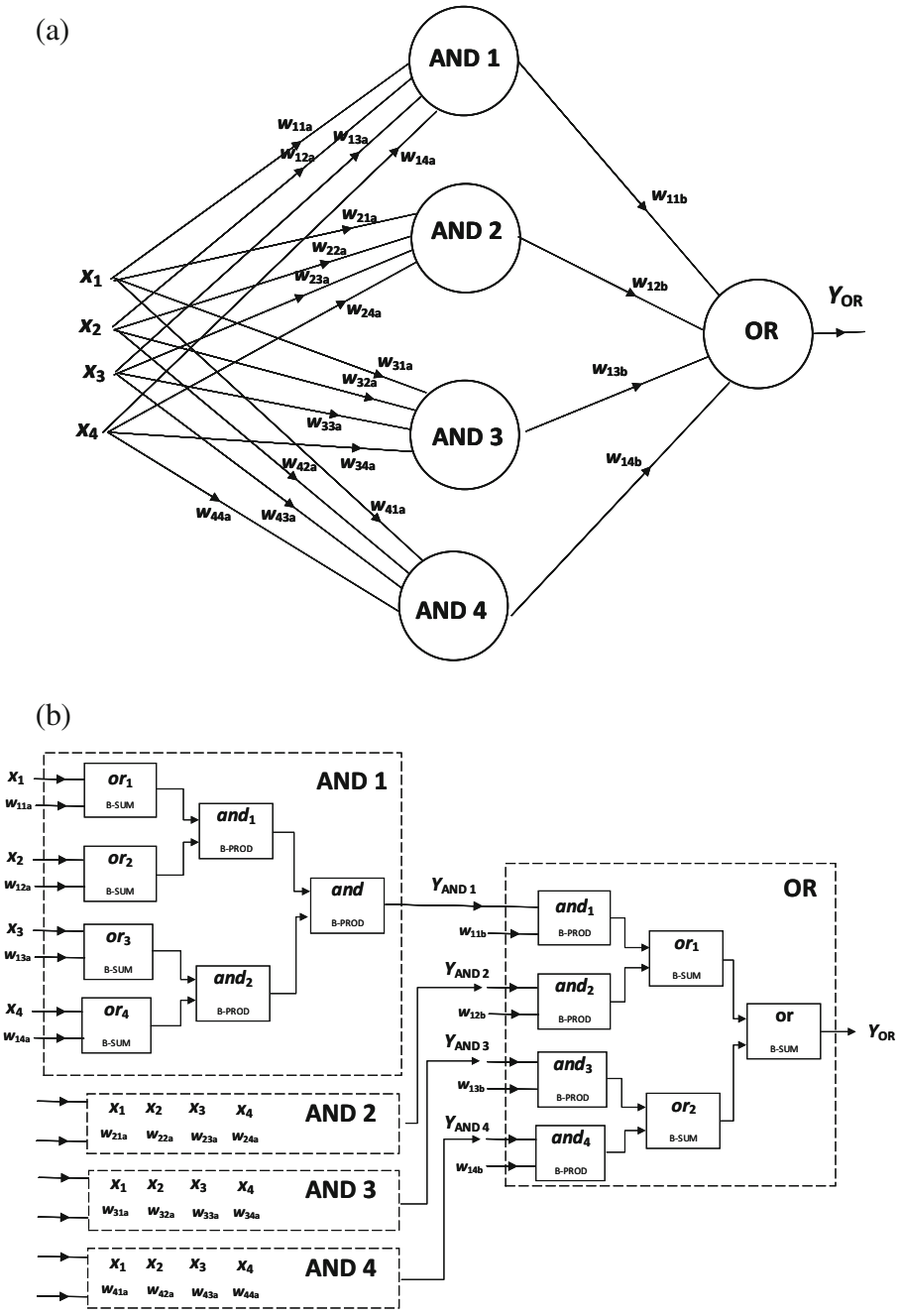

Fig. 9 An example, four inputs, Lukasiewicz AND-OR neural network: **a** a general structure of the network, **b** a more detailed view with a division into particular neurons

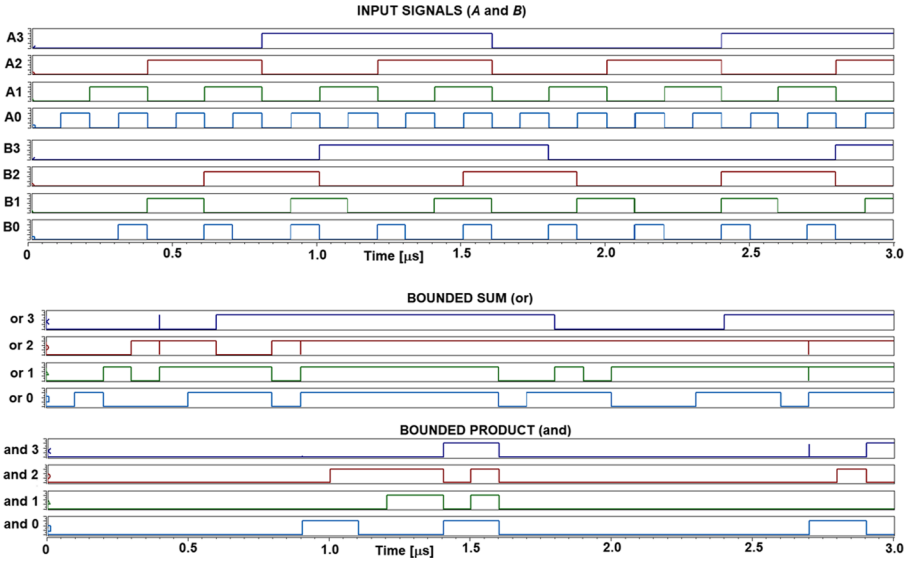

Fig. 10 Selected simulation results of the fuzzy *or* and *and* operators

2.3 Fuzzy logic network OR-AND and AND-OR

Based on the proposed OR and AND type neurons, it is possible to build a fuzzy logic Lukasiewicz OR-AND and AND-OR NN. Example networks of these types are presented in Figs. 7 and 8, respectively.

The proposed network composed of previously described circuits operates fully asynchronously. The signals are provided to its inputs in parallel, without using any clock generator. The data processing rate itself depends on the number of the inputs. It can be noticed, while comparing Figs. 8 and 9. If the number of the inputs increases, the number of the operations in each neuron performed serially also increases. This is visible in the increased number of layers in the binary tree. However, as described earlier, the data processing rate decreases moderately with increasing number of inputs.

The size of the LNN depends on the realized task. When the number of the inputs increases, the neurons with larger number of inputs have to be used. An example implementation of the 4-input AND-OR NN is shown in Fig. 9.

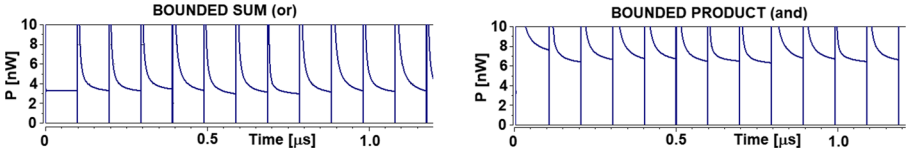

Fig. 11 Power dissipation for the fuzzy *or* and *and* operators

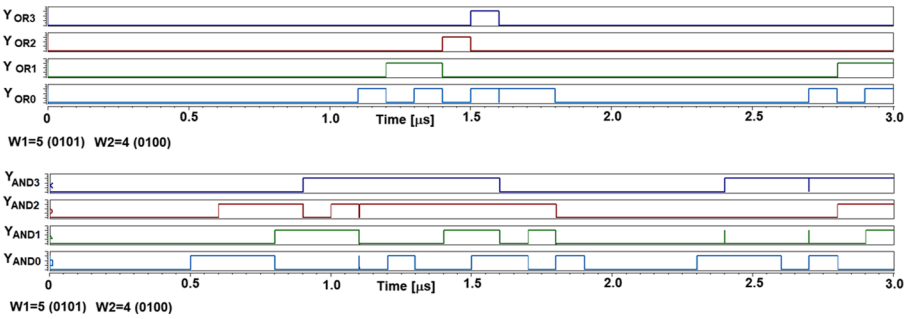

Fig. 12 Performance of the OR (top) and the AND (bootom) Lukasiewicz neuron

3 Results—transistor level verification of the proposed systems

Verification of the proposed circuit has been performed in the HSpice environment in the TSMC 130-nm CMOS technology. To check if the results are reliable, a full corner analysis has been performed, in which the proposed circuits was tested for different values of the process, voltage, and temperature (PVT) parameters. A series of performed tests covered typical (TT), fast (FF), and slow (SS) transistor models, temperatures varying in-between −40 and 100 °C and different values of the supply voltage (from 1.2 to 1.5 V). The difference between the worst case (SS/1.2V) and the best case (FF/1.5V) scenarios was equal to about 140%; however, the circuits worked properly in each case.

Figure 10 presents selected results for the Bounded Product (*and*) and the Bouded Sum (*or*) operations. The signal resolutions in the tests were equal to 4 bits, which means that the value of $\mathbb{1}$ was 0xF.

The upper panel of Fig. 10 presents the input signals (A and B), while the bottom diagram the output signals of the operators. The simulations were performed for V_{DD}=1.5V and T=20 °C.

Figure 11 presents the power dissipation for particular operators. In case of the BP operator (*and*) the power dissipation approximately equals 7 nW, while in case of the BS operator (*or*) it approximately equals 4 nW.

The difference results from the structure of both operators. Note (Figs. 1 and 3) that for the *and* operator, two adders are used, while for the *or* operator only one.

Figure 12 presents simulations of the fuzzy OR and AND Lukasiewicz neurons. The input signals for all simulations are the same, as in Fig. 10. For a better clarity, the x_1 signal has been changed to the A signal, whereas the x_2 signal to B. Since the output signals are 4-bit signals, the a_3, a_2, a_1, a_0 are particular bits of the A signal,

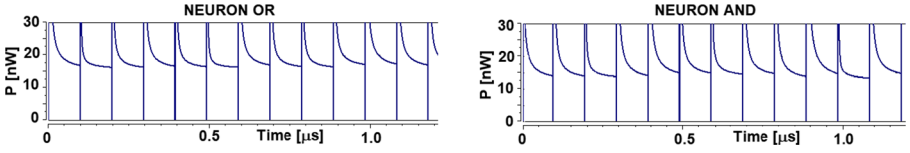

Fig. 13 The power dissipated for the the fuzzy neuron OR and AND

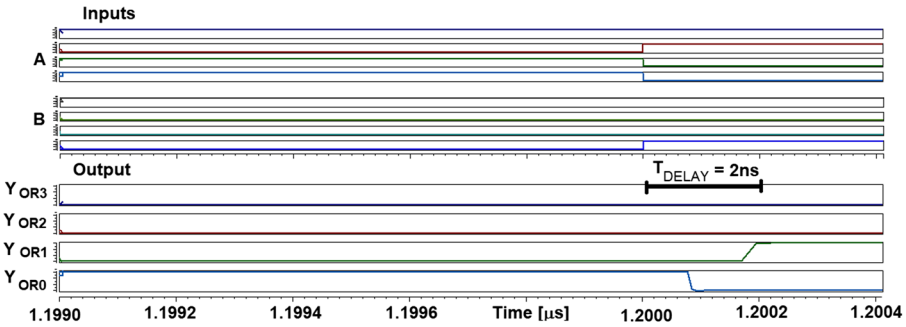

Fig. 14 The delay time at the output of the OR neuron

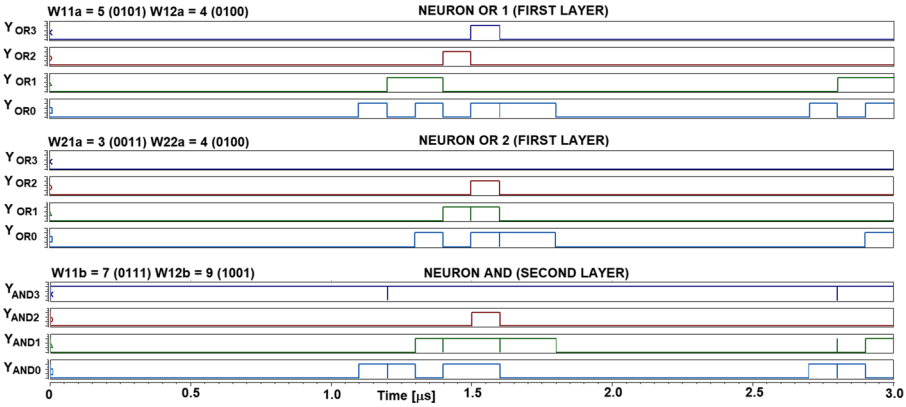

Fig. 15 Simulation work of the Fuzzy OR-AND Network

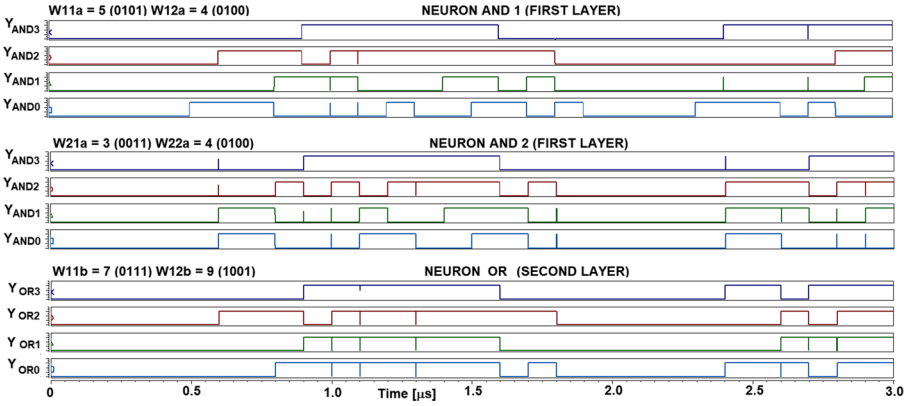

Fig. 16 Simulation work of the Fuzzy AND-OR Network

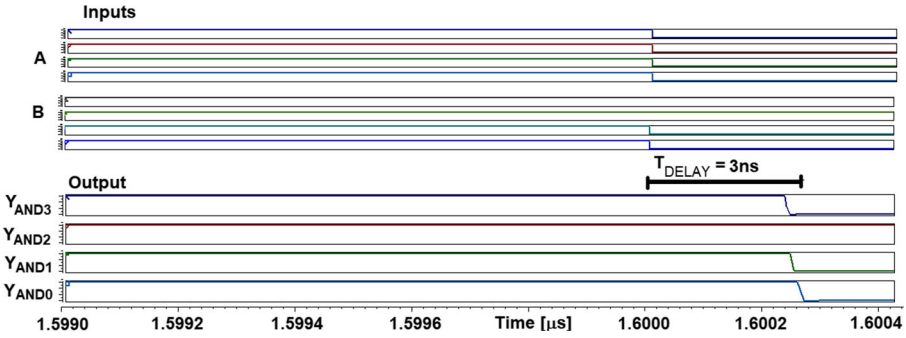

Fig. 17 The signal delay time at the AND-OR network output

with the a_3 being the most significant bit. The same applies to the B signal. The neuron weights in both cases have been set in the same way, i.e., w_1=5 (0101), w_2=4 (0100). As can be observed, both neurons operate properly.

Figure 13 presents the power dissipation for both neurons. The power dissipation in the case of an OR neuron is approximately 17 nW, while for the AND neuron approximately 15 nW.

A slight difference results from the structure of both neurons and the use of particular operators in them. In case of the OR neuron, we use two operators: *and* and one *or*, while for the AND neuron two *or* operators and one *and* operation. As shown earlier, the *and* operator dissipates more power than the *or* one.

The signal at the output of the OR neuron appears approximately 2 ns after changing the signals at its input, as shown in Fig. 14. This means that the neuron works at a frequency of about 500 MHz. The simulations were performed at the supply voltage of V_{DD} = 1.5V and at 20 °C. In case of the AND neuron, the reaction time of the system is similar.

Figures 15 and 16 present the simulations of two-layer Lukasiewicz OR-AND and AND-OR neural networks.

To more accurately illustrate the processes occurring inside the network, we present both the output signal and the signals at the outputs of particular neurons, i.e., in the first layer of the NN. The values of the weights for both NNs have the same values, i.e., w_{11a}= 5 (0101), w_{12a} = 4 (0100), w_{21a}=3 (0011), w_{22a} = 4 (0100), w_{11b} = 7 (0111), w_{12b} = 9 (1001). In both cases, the network works correctly. The time delay appearing at the output of the AND-OR network in relation to the changes of its input signals equals 3 ns, as shown in Fig. 17.

4 Conclusion

Novel transistor-level implementations of fuzzy logic neurons and networks suitable for very fast, low power data processing has been presented in the paper.

The proposed circuits operate fully asynchronously, which means that no clock generator is required to perform particular fuzzy logic operations. The circuits

can be used in larger fuzzy systems, working in parallel, in which larger chains of such operators also operate fully asynchronously. The synchronization is still possible, if necessary, by the use of switches controlled by a clock generator. However, the synchronization may be applied at the ends of larger chains, which is an advantage here, as it simplifies the control of such realized larger fuzzy systems.

The presented circuits (operators, neurons and networks) have been designed and verified in the CMOS 130-nm technology. They offer data processing rates at the level of even several Msamples/s. These parameters can be substantially improved if the circuits are redesigned in newer CMOS technologies.

One of basic (atom) building blocks here are the 1-bit full adders that are components of larger multi-bit full adders. Recently, we developed a prototype chip, in which such full adders as the ones used here have been implemented and verified by means of laboratory measurements.

References

1. Tan, Q., Wei, Q., Hu, J., Aldred, D.: Road vehicle detection using fuzzy logic rule-based method. In: International Conference on Fuzzy Systems and Knowledge Discovery, pp. 3 (2010)
2. Sharma, K., Kumar Palwalia, P.: A modified PID control with adaptive fuzzy controller applied to DC motor. In: International Conference on Information, Communication, Instrumentation and Control (ICICIC) (2017)
3. Chen, Z., Gomez, S.A., McCormick, M.: A fuzzy logic controlled power electronic system for variable speed wind energy conversion systems. In: International Conference on Power Electronics and Variable Speed Drives (2000)
4. Sreedivya, K.M., Aruna Jeyanthy, P., Devaraj, D.: Fuzzy logic based power system stabilizer for damping low frequency oscillations in power system. In: International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICEEIMT) (2017)
5. Seker, H., Odetayo, M.O., Petrovic, D., Naguib, R.N.G.: A fuzzy logic based-method for prognostic decision making in breast and prostate cancers. IEEE Trans. Inf. Technol. Biomed. **7**, 2 (2003)
6. Cetin, O., Kurnaz, S., Kaynak, O.: Fuzzy logic based approach to design of autonomous landing system for unmanned aerial vehicles. J. Intell. Robot. Syst. **61**, 239–250 (2011)
7. Jin, M., Zhao J., Jin J., Yu G., Li W.: The adaptive Kalman filter based on fuzzy logic for inertial motion capture system. Measurement, Elsevier **7**, 196–204 (2014)
8. Banach, M., Wasilewska, A., Długosz, R., Pauk, J.: Novel techniques for a wireless motion capture system for the monitoring and rehabilitation of disabled persons for application in smart buildings. Technol. Health Care, IOS Press **26**(S2), 671–677 (2018)
9. Milanés, V., Villagrá, J., Godoy, J., Simó, J., Pérez, J., Onieva, E.: An intelligent V2I-Based traffic management system. IEEE Trans. Intell. Transp. Syst. **1**(49-58), 13 (2012)
10. Salman, M.A., Ozdemir S., Celebi F.V.: Fuzzy traffic control with vehicle-to-everything communication, Sensors, https://doi.org/10.3390/s1802036827, (368) (2018)

11. Banach, M., Długosz, R.: Real-time locating systems for smart city and intelligent transportation applications. In: IEEE 30th International Conference on Microelectronics (Miel 2017) (231-234) (2017)
12. Li, T.H.S., Chen, Ch.-Y., lim, K.-CH.: Combination of fuzzy logic control and back propagation neural networks for the autonomous driving control of car-like mobile robot systems. In: Proceedings of SICE Annual Conference (2010)
13. Kayacan, E., Kayacan, E.L., Ramon, H., Saeys, W.: Adaptive Neuro-Fuzzy control of a spherical rolling robot using Sliding-Mode-Control-Theory-Based online learning algorithm. IEEE Transactions on Cybernetics **43**, 1 (2013)
14. Allah Hooshmand, R., Parastegari, M., Forghani, Z.: Adaptive neuro-fuzzy inference system approach for simultaneous diagnosis of the type and location of faults in power transformers. IEEE Electr. Insul. Mag. **28**, 5 (2012)
15. Yen, J., Langari, R., Zadeh, L.A.: Industrial Applications of Fuzzy Logic and Intelligent Systems. IEEE Press, New York (1995)
16. Nagaraj, R., Mayurappriyan, P.S., Jerome, J.: Microcontroller based fuzzy logic technique for dc-dc converter. In: International Conference on Power Electronics (2006)
17. Rudas, I.J., Batyrshin, I.Z., Hernández Zavala, A., Camacho Nieto, O., Horváth, L., Villa Vargas, L.: Generators of fuzzy operations for hardware implementation of fuzzy systems, advances in artificial intelligence. In: 7th Mexican International Conference on Artificial Intelligence (MICAI) (2008)
18. Meisam Ramzanzad. M., Rashidy Kanan, H.: A new method for design and implementation of intelligent traffic control system based on fuzzy logic using FPGA. In: Iranian Conference on Fuzzy Systems (IFSC) (2013)
19. Guo, S., Peters, L., Surmann, H.: Design and application of an analog fuzzy logic controller. IEEE Trans. Fuzzy Syst. **4**, 4 (1996)
20. Yamakawa, T., Miki, T.: The current mode fuzzy logic integrated circuits fabricated by the standard CMOS process. IEEE Trans. Comput. **C-35**, 2 (1986)
21. Długosz, R., Pedrycz, W.: Łukasiewicz fuzzy logic networks and their ultra low power hardware implementation, Neurocomputing, Elsevier, 73 (2010)
22. Sanchez-Solano, S., Barriga, A., Jimenez, C.J., Huertas, J.L.: Design and application of digital fuzzy controllers. In: International Fuzzy Systems Conference (1997)
23. Baturone, I., Sanchez-Solano, S., Barriga, A., Huertas, J.: Implementation of CMOS fuzzy controllers as Mixed-Signal integrated circuits. IEEE Trans. Fuzzy Syst. **5**, 1 (1997)
24. Talaśka, T., Długosz, R., Skruch, P.: Efficient transistor level implementation of selected fuzzy logic operators used in control systems. In: Advances in Intelligent Systems and Computing, Trends in Advanced Intelligent Control, Optimization and Automation, vol. 577. Springer (2017)
25. Talaśka, T.: Implementation of fuzzy logic operators as digital asynchronous circuits in CMOS technology. In: International Conference on Microelectronics (MIEL) (2017)
26. Navi, K., Doostaregan, A., Moaiyeri, M., Hashemipour, O.: A hardware-friendly arithmetic method and efficient implementations for designing digital fuzzy adders. Fuzzy Set. Syst, Elsevier **185**(1), 111–124 (2011)
27. Zavala, A.H., Batyrshin, L.Z., Nieto, O.C., Castillo, O.: Conjunction and disjunction operations for digital fuzzy hardware. Appl. Soft. Comput. **13**, 7 (2013)