Hardware Implementation of the Particle Swarm Optimization Algorithm

Tomasz Talaśka*, Rafał Długosz*†, Witold Pedrycz‡
* UTP University of Science and Technology
Faculty of Telecommunication, Computer Science and Electrical Engineering
ul. Kaliskiego 7, 85-796, Bydgoszcz, Poland
† Delphi Poland S.A.
ul. Podgórki Tynieckie 2, 30-399, Kraków, Poland
‡ University of Alberta
Department of Electrical and Computer Engineering
Edmonton, AB T6G 2V4, Canada
E-mails: talaska@utp.edu.pl, rafal.dlugosz@gmail.com, wpedrycz@ualberta.ca

***Abstract*—In this paper, we present a concept of a transistor level implementation of the Particle Swarm Optimization (PSO) algorithm that belongs to the group of unsupervised learning algorithms aimed at the design of artificial neural networks (ANNs). The algorithm exhibits an ability to search for an optimal solution in a multidimensional data space, in which many sub-optimal solutions may exist. The ANN that operates in accordance with the PSO algorithm is composed of a set of cooperating particles (agents) that explore an input data space and communicate information on the best found solution to other particles. The PSO algorithm is usually implemented in software. We in our investigations focus on its transistor level realization. Such an approach enables parallel data processing, in which the overall data rate only moderately depends on the number of particles. Most of the operations and components of such implemented PSO algorithm may be reused considering our former CMOS realizations of other self-organizing learning algorithms. This allowed us to assess main parameters of the PSO.**

***Keywords*—Particle Swarm Optimization Algorithm, CMOS, Artificial Neural Networks, Analog signal processing**

I. Introduction

The idea of the Particle Swarm Optimization (PSO) algorithm is based on the observation of nature and the behavior of swarms e.g. of ants or bees. In swarms, the welfare of particular particles (or agents) is not as important as the good of the overall swarm, and the behavior of the swarm is a result of the behavior of particular members of the swarm. Particular agents individually gain their own experience and communicate it to other agents, thus facilitating their work. On the other hand, each particle acquires the experience gained earlier by the overall swarm. This approach is one of the best ways of solving various problems. A single agent is a generic processing unit, once it is considered of the computation capabilities, but in the context of the overall group it is able to perform even very complex tasks. For this reason, imitating the behaviour and the intelligence of the flock of birds, the shoal of fish or the swarm of ants is one of very efficient techniques used in the artificial intelligence area. In can be said that the PSO algorithm to some extent mimics also the behaviour of human societies. While gaining and sharing personal experience by particular units, the overall population is gradually moving toward better solutions.

The PSO algorithm was invented in 1995 by Russell Eberhart and James Kennedy [1]. In [1] authors presented a new method for optimization of continuous, non-linear functions based on:

- Artificial life (flocks, shoals, swarms).
- Evolutionary computation (genetic algorithms and evolutionary programming).

The idea of the PSO might sound complex, but it is relatively a simple algorithm. Over a given number of iterations, a set of variables associated with particular agents have their values adjusted to move the agents closer to an optimum solution – a target. For example, let us imagine a flock of birds circling over a more or less unknown area, where they can smell a hidden source of food. The one who is the closest to the food chirps the loudest and remaining birds swing around in this direction. If any other circling bird comes closer to the target, it also chirps louder that gains the signal and thus speeds up the process of finding the target [2]. It is worth to note that birds always return to the place, in which they had found the food earlier. This causes that new particles (new swarm members) quickly acquire the common experience of the swarm.

The movement of particles in the PSO algorithm resembles the movement that takes place in the swarm. However, the particles in the PSO algorithm are abstract entities with in the abstract environment. For this reason, some effects that occur in this case are not present in real swarms. For example, particular units may fall on each other and occupy the same points in the input data space. The mathematical description of the behaviour of the swarm is also only a simplified model of the behaviour of real swarms.

In the PSO algorithm, i^{th}- particle is described by (or has access to) the following data:

- coordinates of its own position, x_i, in the input data space,

- an information on its own velocity, v_i,
- a personal (own) best solution, p^{best}, as well as the information in which place this solution has been found,
- a global (swarm) best solution, g^{best}, with the information in which place this solution has been found,

In addition to these parameters, each particle can obtain the value of the, so called, fitness function for a given position in the input data space. The fitness function is a kind of a target function [3], [4], [5], [6]. In a typical verification process of the PSO algorithm various functions are used as a fitness function (for example: Rosenbrock, Rastrigin, Sphere, ect.) [7], [8], [9], [10].

The parameters are modified in the course of the learning process, in accordance with relatively simple mathematical rules, presented in following Section. The implementation of this algorithm requires using only basic operations, and thus it can be relatively easy implemented at the transistor level.

In this paper we propose a hardware model of the PSO algorithm, suitable for the realization of the algorithm as an application specific integrated circuit (ASIC). In the presented work we focus on its analog realization, as the project is a continuation of our former works on analog, parallel, self-organizing neural networks [11], [12], [13], [14], [15], [16], [17], [18]. The structure of the PSO algorithm allows for a large reuse of the previously developed circuit solutions.

In the literature one can find counterpart digital realizations of the PSO algorithm in the field programmable gates array (FPGA) [7], [8], [9], [10]. The main purpose of such realizations is a parallel operation of particular units (members of the swarm) and continuous changes of their positions and velocities. Sequential software solutions are inefficient due to relatively low data rate, which decreases with the increasing number of particles. The proposed model allows to achieve a parallel data processing, with additionally reduced energy consumption when compared to FPGA realizations.

The paper is organized as follows. In Section II we present the idea as well as the mathematical description of the PSO Algorithm, which is a basis for the hardware model of the algorithm, described in Section III. In Section IV we make some estimates of the influence of the number of particles and the number of dimensionality on the power dissipation and the chip area of the overall PSO. The conclusions are drawn in Section V.

II. Fundamentals of the PSO Algorithm

In the PSO algorithm all particles perform, in the loop, a specific list of tasks, as described below.

Computational sequence in the PSO algorithm:

Initialize a population of particles with random positions and velocities in an n-dimensional data space.

start loop

For each, k^{th}, iteration:

1) Each i^{th} particle in the swarm provides its own position, $x_i^k = \{x_{i,1}, x_{i,2}, ..., x_{i,n}\}^k$, to the fitness function block, which returns a corresponding value of the p_i^k signal for this position.
2) The best value among all p_i^k values is determined by the use of the Min or Max function. Depending on the problem definition –the structure of the fitness function – either the maximum or minimum value is considered as the best one.
3) The best value determined in Step 2 is then compared with a recent global best value, g^{best}, stored in the memory. If the new value is better than the previous one, it replaces the old value. The position for which it has been found is also stored in the memory, replacing the position for which the previous g^{best} value was found in previous algorithm iterations.
4) Each particle compares its own p_i^k value, determined in this iteration, with its own recent local best value, p_i^{best}. If the p_i^k value is better than the p_i^{best} value, the previous value is replaced by the new one. The position for which the p_i^k value has been found is also stored in the local memory of a given particle.
5) On the basis of the determined p_i^{best} and g^{best} values, each particle updates its own velocity v_i^{k+1} and then its own position x_i^{k+1}, according to:

$$v_i^{k+1} = w \cdot v_i^k + c_1 \cdot r_1 \cdot (p_i^{\text{best}} - x_i^k) + \\ + c_2 \cdot r_2 \cdot (g^{\text{best}} - x_i^k) \tag{1}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{2}$$

where:

- x_i^k and x_i^{k+1} are the current and the updated positions of the i^{th} particle, respectively,
- v_i^k and v_i^{k+1} are the current and the updated velocities of the i^{th} particle, respectively,
- r_1 and r_2 are random values coming from a uniform distribution in the range [0,1],
- c_1 is the personal acceleration coefficient – a constant value,
- c_2 is the social acceleration coefficient – a constant value,
- w is the inertia coefficient – a constant value,
- $(p_i^{\text{best}} - x_i^k)$ is a distance vector between the personal best solution and a current position of a given, i^{th}, particle,
- $(g^{\text{best}} - x_i^k)$ is a distance vector between the global best solution and a current position of a given, i^{th}, particle.

6) If a given criterion is met (usually a sufficiently good fitness value or a maximum number of iterations) exit the loop.
   Also one has to make sure that the new position of the particle is located in the range of the search space. This could be done by using a simple limiter.

end loop

Figure 1 illustrates the computational process for an example case of 2-dimensional search space. It shows how the position and the velocity of a single particle are modified.

At this point, it is necessary to clarify one aspect. In the literature it is commonly used the velocity term v. However, in 2 the addition of the velocity to the position may be confusing. In practice, the velocity is a Δx for a given iteration.

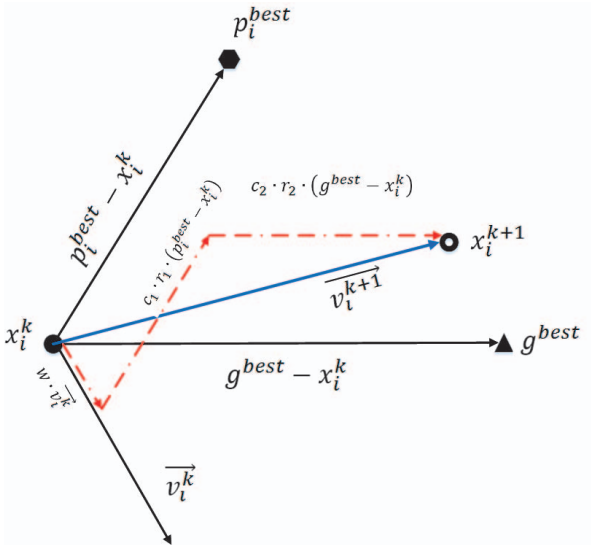

Fig. 1. Graphical presentation of movement of particles in the PSO algorithm.

III. Hardware Model of the PSO Algorithm

In systems based on the PSO algorithm all particles change their positions at the same time. For this reason it is reasonable to implement this algorithm using technologies that support the parallel data processing, such as FPGA platforms or in ASICs.

FPGA platforms facilitate the realization of the PSO algorithm, as in this case the behaviour of even complex system with large number of particles can be described in a hardware description language. This type of implementation also enables the modifications and corrections of the system at any development stage. We pay for it with larger power dissipation and less miniaturization than in case of the ASIC realization. In ASICs, on the other hand, we have to deal with various physical phenomena, especially in case of the analog approach.

Transistor level approach offers more opportunities. One of them is the possibility to mix together analog and digital blocks, which may lead to more flexible solutions. A pure analog implementation is not practical. On the other hand, in a fully digital approach the circuit is usually robust against the process, temperature, voltage variation, as well as the transistor mismatch. However, in this case even basic operations and functions are realized using relatively complex circuit components. For this reason, in this study we focus on a solution with a predominance of analog current-mode components. The current-mode technique facilitates the implementation of such operations as addition, subtraction, multiplication and comparison of signals. Such operations dominate in the PSO algorithm. Circuits of this type usually offer a lower power dissipation than their digital counterparts [11], [18], [19].

One of typical problems with analog realization is a reduced precision that results from the influence of the noise. Analog circuits are also less robust against the physical phenomena described above. One can also mention the leakage effect, relevant in case of using analog memory cells. In the proposed analog approach, however, we do not need to develop and optimize all required components from scratch. A majority of required components have been reused by us from our former projects of analog self-organizing neural networks, verified by laboratory tests [11] - [18].

A. Basic operations and components used in the proposed model of the PSO algorithm.

Major calculation steps of the PSO algorithm have been described in previous Section. Below we present how these operations can be implemented at the transistor level.

1) Calculation of p^{best} and g^{best} signals: One of the problems in the PSO algorithm, not present in NNs previously realized by us, is the necessity of the calculation of the fitness function. In the worst case we assume that the shape of this function is not known in advance, as it depends on a real problem with which the NN has to deal. In this case the p^{best} and g^{best} values returned by this function has to be determined outside the chip on the basis of current positions of particular particles. An analogous situation occurs when a swarm arrives in a new area, the map of which is not yet known.

In the literature, however, one can find various fitness functions [3], [4], [5], [6], [7], [8] which are used to optimize the learning algorithms. Using such functions is reasonable, as it allows for a direct comparison of various implementations of the PSO algorithm. Simple polynomial functions of this type may be implemented at the transistor level, however it makes sense only at the stage of the optimization of the algorithm. Since it is complex problem itself, we leave this issue for future investigations.

2) Updating the velocities and the positions of particles: In the PSO algorithm the important information are positions of particular particles in which these units reached their best solutions (local extremes), p^{best}, as well as the position for which a global extreme, g^{best}, has been found. The coordinates of these positions have similar meaning as neuron weights in conventional NNs. For this reason, the coordinates are updated in a similar way, as the neural weights are adapted in a typical NN.

The calculation of the updates of the velocities (Δx factor) and the positions of particulr particles require several basic operations, schematically shown in Fig.2. This block contains two summing, two subtracting and two multiplication blocks. Since in the proposed model all intermediate signals are currents, we perform the summation and subtraction operations simply in junctions, according to first Kirchhoff's law.

The r, c and w signals are constants that in the proposed solution are represented by multi-bit digital signals. Theoretically, they are floating-point numbers with the values in the range [0, 1]. In the proposed implementation they are fixed point numbers, with the maximum values dependent on an assumed resolution. The $r_1 \cdot c_1$ and $r_2 \cdot c_2$ terms are kept in the memory as single constants. The multiplication by these terms is performed by the use of multi-output current mirrors, with the widths of particular transistors being following powers of 2.

The updated values of the x and the v terms substitute their previous values stored in analog memory block associated with each particle. The circuits that are used to update the analog memory cells are the same, as in the adaptation block used by us in one of our former projects [14]. The same circuit is also used to update the values of the p^{best} and g^{best} signals in the memory.

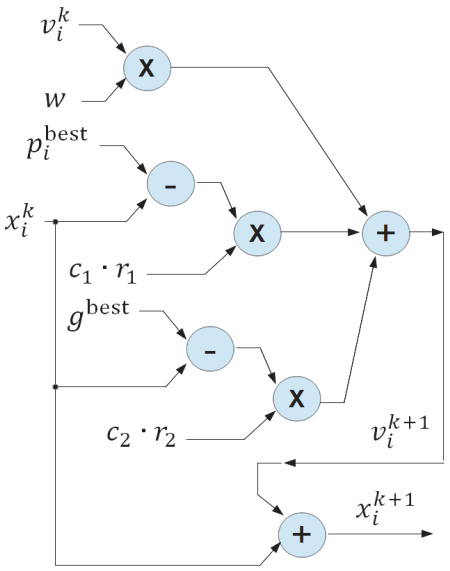

Fig. 2. Block diagram of the proposed hardware model of the PSO algorithm – the part responsible for the calculation of the v_i^{k+1} and the x_i^{k+1} terms.

3) Searching for the best value among the p^{best} signals: This operation can be carried out by the use of a conventional Min or Max circuit, similar to those that are used in nonlinear filters. We perform this operation by the use of a binary tree (BT) winner selection circuit (WC), whose simplified version is shown in Fig. 3. The circuit is composed of MIMA2 blocks, shown in Fig. 4, that determine a single maximum or minimum value of two input signals. The core block of this circuit is a current-mode comparator, shown in Fig. 5. To improve the accuracy of the circuit we used cascoded current mirrors, visible in Fig. 5.

In BT circuits the best value is determined in steps at particular layers of the tree. At the first layer particular p_i^{best} values directly compete in pairs. The winner from each pair is allowed to take part in the competition at the second layer. This scheme is repeated up to the last layer of the tree, at which a single best signal is distinguished. We used the BT circuit since in the scope of our interest is not only the value of the best solution, but also the address of the particle that is the owner of this solution.

In real application we can use one of the WSCs proposed by us in [13] and [17]. In both these circuits we implemented mechanisms that prevent the accumulation of errors, that in conventional BT WSCs are introduced at particular layers of the tree.

To check if the best p_i^{best} value determined by the WSC is better than the previously determined g^{best} value we use the UB block shown in Fig. 3 (bottom). The comparator compares both these signals and updates the memory accordingly.

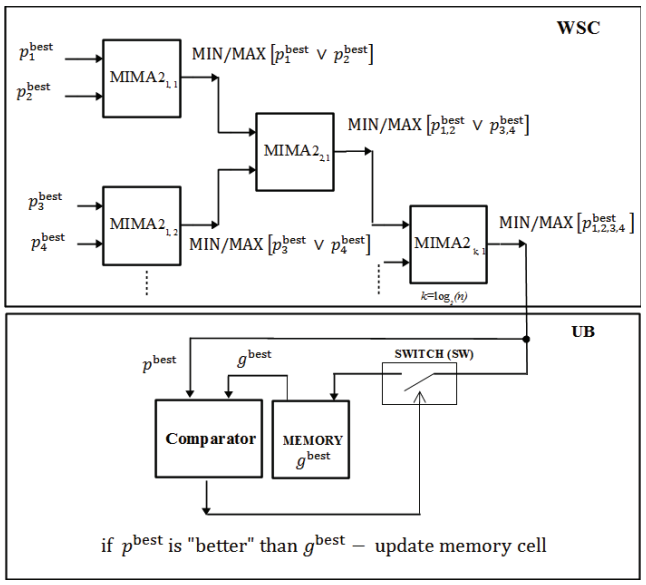

Fig. 3. Block diagram of the proposed hardware model of the algorithm PSO – the part responsible for the determination of the p^{best} and g^{best} values.

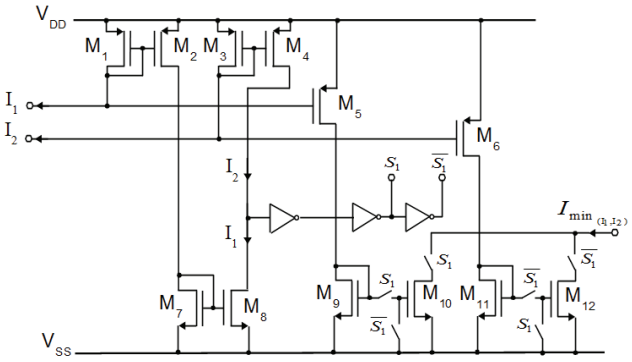

Fig. 4. The MIMA2 circuit (Min or Max function) used in the proposed hardware model of the PSO algorithm.

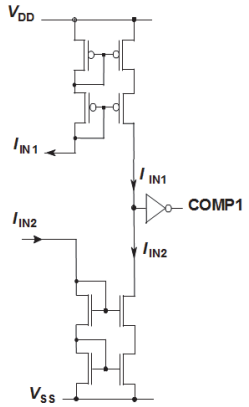

Fig. 5. Current-mode comparator used in the proposed hardware model.

IV. Estimation of Basic Parameters of the Proposed Hardware Model

The proposed model is currently under development. Since most of the components were already implemented at the transistor level and verified by means of measurement tests, we have reliable estimates of such parameters as the hardware complexity and the power dissipation of these components. In this work, we present the estimates for the overall PSO algorithm with addition of the blocks that are new in this case. We do not present simulation or measurement results as detailed results are presented in our former works. The results shown in Fig. 6 are for the CMOS TSMC 180 nm technology, in which our previous projects were realized. The values of both parameters are presented as a function of the number of particles in the swarm and the number of coordinates, n, in the input data space.

On the basis of the former results it can be shown that data rate at the level from 0.5 to 1.5 MSamples/s is possible to reach. These values moderately depend on the number of particles. In the proposed solution each particle operates in parallel. The largest influence of size of the network is visible in WSC block, as the number of layers in the tree increases with the number of the calculation units.

V. Conclusions

In the paper we present a hardware model of the Particle Swarm Optimization (PSO) algorithm. In our investigations we aim at a fully custom implementation of this algorithm at the transistor level. To our best knowledge such solutions were not presented in the literature so far. The existing hardware realizations are those on FPGA platforms. This shows that there is a room for novelties in this area.

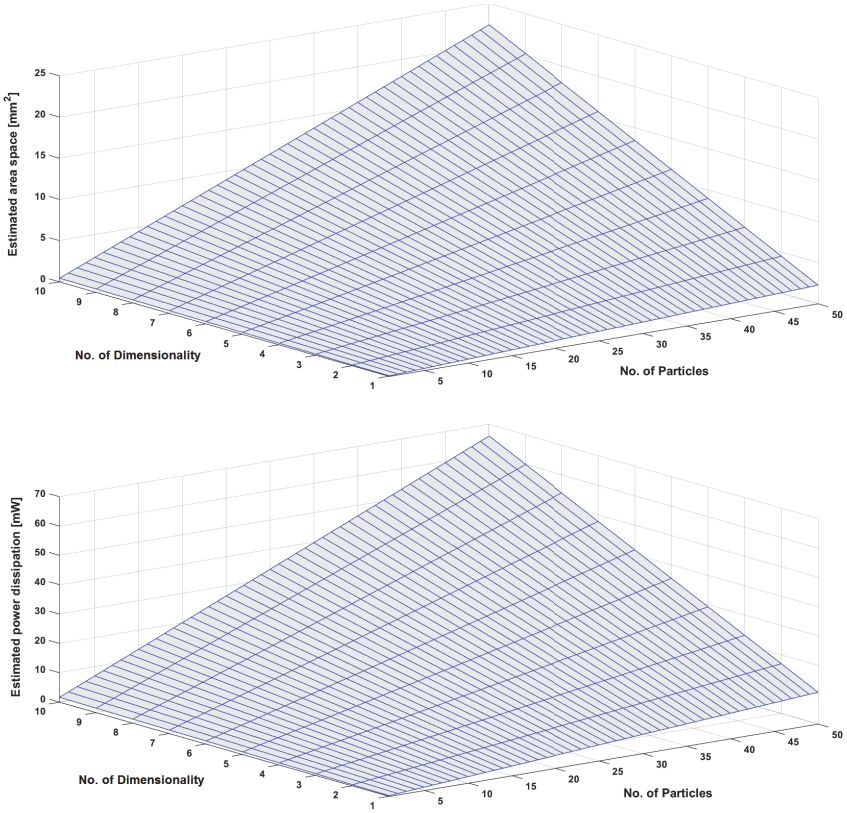

Fig. 6. Estimates of the: (top) chip area, (bottom) power dissipation as a function of the number of particles and the number of coordinates of x positions of particulr particles.

The proposed model is mostly composed of analog components, whose design is usually challenging. One of the advantages of this solution is that most of the components of the algorithm may be reused, usually after a small modification, from previous projects.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *IEEE International Conference on Neural Networks*, 1995

[2] K. Thumar,D. Bosamiya, "Evaluate spam detection using hybrid technique of Support Vector Mechine", *International Journal of Engineering Development and Research (IJEDR)*, Vol. 3, Iss. 4, 2015 Available:http://www.ijedr.org/papers/IJEDR1504143.pdf

[3] J. He, T. Chen and X. Yao, "On the Easiest and Hardest Fitness Functions", *IEEE Transactions on Evolutionary Computation*, Vol. 19, Iss. 2, 2015

[4] R. Jiang;, H. Wang, S. Tian, B. Long, "Multidimensional Fitness Function DPSO Algorithm for Analog Test Point Selection", *IEEE Transactions on Instrumentation and Measurement*, Vol. 59, Iss. 6, 2010

[5] A. L. Nelsona, G. J. Barlowb, L. Doitsidisc, "Fitness functions in evolutionary robotics: A survey and analysis", *Robotics and Autonomous Systems*, Elsevier, Vol. 57, Iss. 4, 2009

[6] M. Davarynejad, M. R. Akbarzadeh, N. Pariz, "A novel general framework for evolutionary optimization: Adaptive fuzzy fitness granulation", *IEEE Congress on Evolutionary Computation*, 2007

[7] A. Rathod and R.A. Thakker, "FPGA Realization of Particle Swarm Optimization Algorithm using Floating Point Arithmetic", *International Conference on High Performance Computing and Applications (ICHPCA)*, 2014

[8] R. M. Calazan, N. Nedjah, and L. M. Mourelle, "A hardware accelerator for Particle Swarm Optimization", *Elsevier Journal of Applied Soft Computing*, Vol. 14, Part C, 2014

[9] D. M. Munoz, C. H. Llanos, L. D. Coelho, and A M. Rincon, "Comparison between two FPGA implementations of the Particle Swarm Optimization algorithm for high-performance embedded applications",, *IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, 2010

[10] G. S. Tewolde, D. M. Hanna, R. E. Haskell, "Multi-swann parallel PSO: Hardware implementation", *IEEE Symposium on Swarm Intelligence (SIS)*, 2009

[11] T. Talaśka, M. Kolasa, R. Długosz and W. Pedrycz, "Analog Programmable Distance Calculation Circuit for Winner Takes All Neural Network Realized in the CMOS Technology", *IEEE Transactions on Neural Networks and Learning System*, Vol. 27, No. 3, 2016

[12] T. Talaśka, M. Kolasa, R. Długosz and A.P Farine, "An efficient initialization mechanism of neurons for Winner Takes All Neural Network implemented in the CMOS technology", *Applied Mathematics and Computation, Elsevier*, Vol. 267, 2015

[13] R. Długosz and T. Talaśka, "Low Power Current-Mode Binary-Tree Asynchronous Min / Max Circuit", *Microelectronics Journal, Elsevierl*, Vol. 41, 2010

[14] R. Długosz, T. Talaśka, and W. Pedrycz, "Current-Mode Analog Adaptive Mechanism for Ultra-Low-Power Neural Networks", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 58, No.1, 2011

[15] T. Talaśka, and R. Długosz, "Analog, Parallel, Sorting Circuit for the Application in Neural Gas Learning Algorithm Implemented in the CMOS Technology", *Applied Mathematics and Computation, Elsevier*, 2017, DOI: 10.1016/j.amc.2017.02.030

[16] R. Długosz, T. Talaśka, "A Power-Efficient, Current-Mode, Binary-Tree Min / Max Circuit for Kohonen Self-Organizing Feature Maps and Nonlinear Filters", Electrical Review Journal, R. 86, 11a/2010, 2010

[17] R. Długosz, A. Rydlewski, T. Talaśka, "Novel, low power, nonlinear dilatation and erosion filters realized realized in the CMOS technology", FACTA UNIVERSITATIS, Series: Electronics and Energetics Vol. 28, No 2, 2015

[18] R. Długosz, T. Talaśka, R. Wojtyna and W. Pedrycz, "Realization of the Conscience Mechanism in CMOS Implementation of Winner-Takes-All Self-Organizing Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 21, No. 6, 2010

[19] J.O. Klein, L. Lacassagne, H. Mathias, S. Moutault, A. Dupret, "Low power image processing: analog versus digital comparison", *International Workshop on Computer Architecture for Machine Perception (CAMP'05)*, 2005