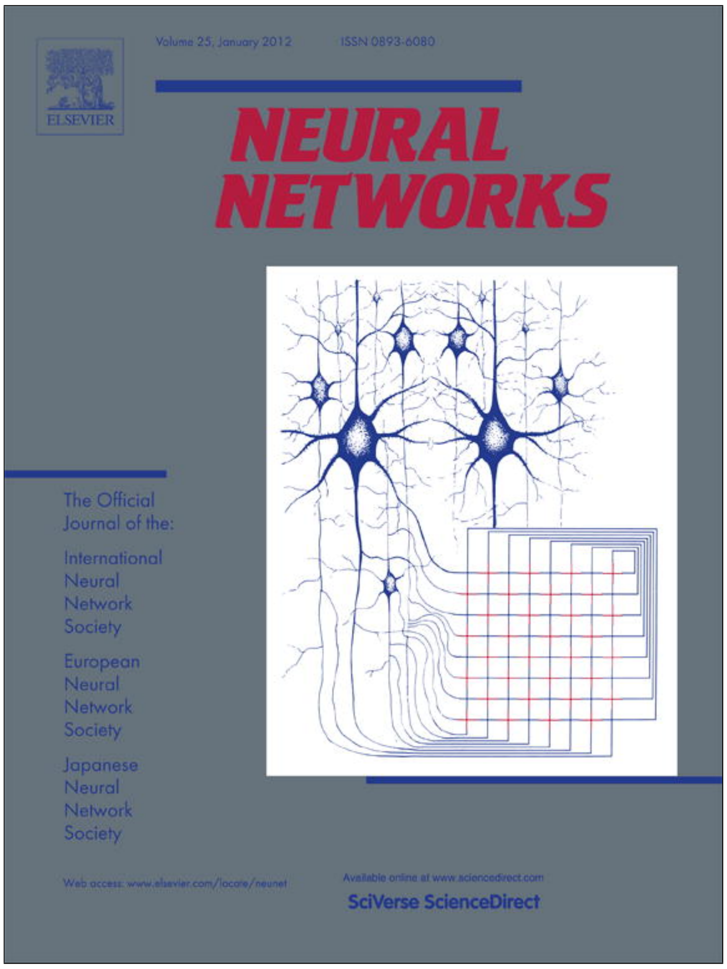
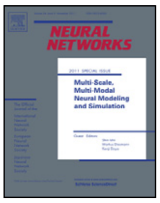

Volume 25, January 2012
ISSN 0893-6080
ELSEVIER
NEURAL NETWORKS
The Official Journal of the:
International Neural Network Society
European Neural Network Society
Japanese Neural Network Society
Web access: www.elsevier.com/locate/neunet
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Neural Networks 25 (2012) 146–160

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



A programmable triangular neighborhood function for a Kohonen self-organizing map implemented on chip

Marta Kolasa[a], Rafał Długosz[a,b,c,*], Witold Pedrycz[d,e], Michał Szulc[f]

[a] University of Technology and Life Sciences, The Faculty of Telecommunications and Electrical Engineering, Kaliskiego 7, 85-796, Bydgoszcz, Poland
[b] Swiss Federal Institute of Technology (EPFL), Institute of Microtechnology, Electronics and Signal Processing Laboratory (ESPLAB), A.L.Breguet 2, CH-2000, Neuchatel, Switzerland
[c] Mars Society Polska, Space Research Center of the Polish Academy of Sciences, Bartycka 18A, 00-716 Warsaw, Poland
[d] University of Alberta, Department of Electrical and Computer Engineering, ECERF Building, Edmonton, AB T6G 2V4, Canada
[e] Systems Research Institute, Polish Academy of Sciences, Warsaw, Newlska 6, Poland
[f] Poznan University of Technology, Chair of Computer Engineering, ul. Piotrowo 3A, 60-965, Poznan, Poland

ARTICLE INFO

Article history:
Received 29 January 2011
Received in revised form 16 July 2011
Accepted 1 September 2011

Keywords:
Neighborhood mechanism
SOM
Parallel data processing
CMOS realization
Low energy consumption
Asynchronous circuits

ABSTRACT

An efficient transistor level implementation of a flexible, programmable Triangular Function (TF) that can be used as a Triangular Neighborhood Function (TNF) in ultra-low power, self-organizing maps (SOMs) realized as Application-Specific Integrated Circuit (ASIC) is presented. The proposed TNF block is a component of a larger neighborhood mechanism, whose role is to determine the distance between the winning neuron and all neighboring neurons. Detailed simulations carried out for the software model of such network show that the TNF forms a good approximation of the Gaussian Neighborhood Function (GNF), while being implemented in a much easier way in hardware. The overall mechanism is very fast. In the CMOS 0.18 μm technology, distances to all neighboring neurons are determined in parallel, within the time not exceeding 11 ns, for an example neighborhood range, *R*, of 15. The TNF blocks in particular neurons require another 6 ns to calculate the output values directly used in the adaptation process. This is also performed in parallel in all neurons. As a result, after determining the winning neuron, the entire map is ready for the adaptation after the time not exceeding 17 ns, even for large numbers of neurons. This feature allows for the realization of ultra low power SOMs, which are hundred times faster than similar SOMs realized on PC. The signal resolution at the output of the TNF block has a dominant impact on the overall energy consumption as well as the silicon area. Detailed system level simulations of the SOM show that even for low resolutions of 3 to 6 bits, the learning abilities of the SOM are not affected. The circuit performance has been verified by means of transistor level Hspice simulations carried out for different transistor models and different values of supply voltage and the environment temperature — a typical procedure completed in case of commercial chips that makes the obtained results reliable.

1. Introduction

Self-organizing Kohonen neural networks (KNNs) belong to the category of networks trained in an unsupervised mode (Boniecki, 2005; Kohonen, 2001; Masmoudi, Dieng, & Masmoudi, 2002). Networks of this type are aimed at the discovery of dependences in highly dimensional spaces. Given this feature, they are often referred to as self-organizing maps (SOMs). Such networks can be realized in various ways. Predominantly, we encounter their software implementations. Another attractive alternative, whose usage is still quite limited, deals with a hardware realization implemented, for example, in the form of Very Large Scale of Integration (VLSI) Application Specific Integrated Circuits (ASIC). This second approach requires solving specific problems of electronic nature, and as such it becomes much more challenging. On the other hand, transistor level implemented SOMs offering a fully parallel operation of all neurons, are much faster than their software counterparts, additionally consuming significantly less power (Abuelma'ati & Shwehneh, 2006; Długosz, Kolasa, & Pedrycz, 2010; Długosz, Talaśka, Pedrycz, & Wojtyna, 2010; Li, Chang, & Siek, 2009; Masmoudi et al., 2002). The low power dissipation results from the possibility to optimize the structure of the SOM directly for a given task that enables a significant reduction of the number of transistors on the chip.

* Corresponding author at: University of Technology and Life Sciences, The Faculty of Telecommunications and Electrical Engineering, Kaliskiego 7, 85-796, Bydgoszcz, Poland. Tel.: +48 668160217.
E-mail addresses: markol@utp.edu.pl (M. Kolasa), rafal.dlugosz@gmail.com (R. Długosz), pedrycz@ece.ualberta.ca (W. Pedrycz), szulc.michal@gmail.com (M. Szulc).

0893-6080/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2011.09.002

Although parallel data processing is also possible in Field Programmable Gate Arrays (FPGAs) and recently in Graphics Processing Units (GPUs) however it happens at the expense of very high power dissipation. Ultra low power dissipation as well as low chip area available in case of the ASIC implementation open new possibilities of applications of such networks e.g., in a variety of low power portable devices. A very promising is the ability to use such SOMs in Wireless Sensor Networks (WSNs) or Wireless Body Area Networks (WBANs) in medical diagnostics. In a typical WSN, a set of small, ultra low power sensors collect various data, transmitting them to a base station for further analysis. In case of the WBAN, the sensors located on the human body collect biomedical data. In systems of this type, most of the collected data are typically transmitted to a base station, which creates the problem of large energy consumption, as the radio-frequency (RF) communication modules of the sensors consume even 95% of total energy. This reduces the battery life span (Akyildiz, Su, Sankarasubramaniam, & Cayirci, 2002; Bereketli & Akan, 2009). One of the ways to minimize this problem relies on the optimization of the RF block of particular sensors (Bereketli & Akan, 2009) or the architecture of the WSN (Dubois et al., 2009). Another possibility, which is the objective of the presented work, is to reduce significantly the amount of data being sent to the base station. In this case, some data processing tasks have to be located directly at the sensor level. This concept requires the development of energy efficient and low chip area signal processing blocks (Corbishley & Rodriguez-Villegas, 2007). The proposed low power SOM fits this concept, offering even four orders of magnitude better data rate to energy consumption ratio than a SOM implemented on a typical PC.

We have been working on hardware realized self-organizing NNs for many years, developing a fully analog Winner Takes All (WTA) network, as discussed in Długosz, Talaśka et al. (2010). The recent work in this field (Długosz & Kolasa, 2008; Kolasa & Długosz, 2008) concerns a development of a new flexible and programmable neighborhood mechanism to be used in the analog WTA network described in Długosz, Talaśka et al. (2010), thus significantly increasing its functionality, or in fully digital Kohonen SOM. The proposed mechanism is very fast. It comes as an asynchronous solution that features a very simple structure, as the controlling clock circuitry is not required. The realization of such mechanism requires solving two problems that must be clearly distinguished. The type of the neighborhood function (NF) and its hardware implementation is one of them, important aspect is how to determine quickly in parallel distances between the winning neuron and all its neighbors. In case of the hardware realization both these aspects require two different circuits. In the literature, one can find many solutions for the NF block, mostly analog, but there is a clear lack of efficient solutions of distance determination blocks (DD). The only programmable circuit of this type has been proposed by Peiris (1994). This analog circuit offers a very simple structure with only several transistors per neuron, but it suffers from several disadvantages. The most important of them are: a relatively high power dissipation, low data rate, sensitivity of the parameters to the environment temperature and a mismatch effect present between particular transistors. Some other circuits of this type have been proposed (Li et al., 2009; Macq, Verleysen, Jespers, & Legat, 1993) but they are suitable for small neighborhood range of only one. The shortage of efficient DD circuits was the motivation behind developing our own solution of such circuit. This fully programmable asynchronous digital circuit, in which the neighborhood range is controlled in a wide range, has been described in details in Długosz and Kolasa (2008) and Kolasa and Długosz (2008). In this paper, we focus on the hardware implementation and the optimization of the NF block itself. Nevertheless, as the new NF block is the component of the overall neighborhood mechanism, the key aspects of this concept are briefly presented in next section.

In the Kohonen Winner Takes Most (WTM) SOM, the adaptation process of the weights (connections) of the neurons is expressed as follows:

$$W_j(l+1) = W_j(l) + \eta(k)G(R, d(i,j))[X(l) - W_j(l)] \tag{1}$$

where $\eta(k)$ is the learning rate present in the kth training epoch, W_j are the weight vectors of the corresponding particular neurons in the map, while X is an input training pattern present in an lth cycle. These neurons that belong to the winner's neighborhood, are trained at different intensities that depend on the neighborhood function $G()$, which depends on the distance $d(i,j)$, between the winning ith neuron and the other jth neurons in the map. In the classical approach, a simple Rectangular Neighborhood Function (RNF) is used (Boniecki, 2005; Kohonen, 2001):

$$G(R, d(i,j)) = \begin{cases} 1 & \text{if } d(i,j) \le R \\ 0 & \text{if } d(i,j) > R \end{cases} \tag{2}$$

where $d(i,j)$ is a distance between the winning, ith, neuron and some other, jth, neuron in the map that belongs to the winner's neighborhood, while R is the range of the neighborhood that is decreased after each epoch. At the beginning of the learning process R equals an $R_{\max}$ value that in this paper is one of the parameters that is optimized. The commonly encountered opinion is that much better learning results can be achieved if the Gaussian Neighborhood Function (GNF) is used instead of the rectangular one (Mokriš & Forgáč, 2004). Let us recall that the Gaussian function is defined as follows:

$$G(R, d(i,j)) = \exp\left(-\frac{d^2(i,j)}{2R^2}\right). \tag{3}$$

Different hardware realizations of the Gaussian function have been proposed (Abuelma'ati & Shwehneh, 2006; Li et al., 2009; Masmoudi et al., 2002). The existing implementations usually involve analog circuits. On the other hand, since our proposed neighborhood mechanism is to be used in either the analog or the digital SOM therefore the objective was to develop an efficient digital NF circuit. The Gaussian function, due to its complexity, resulting from squaring, division and the exponential operations, is difficult to be realized given low chip area and low power hardware. This was one of the motivations to verify whether the Triangular Neighborhood Function (TNF) can be used as a substitute for GNF. This function is defined as:

$$G(R, d(i,j)) = \begin{cases} -a(\eta_0)\cdot\Big(R - d(i,j)\Big) + c & \text{if } d(i,j) \le R \\ 0 & \text{if } d(i,j) > R \end{cases} \tag{4}$$

where $a()$ is the assumed steepness of this function, η_0 is the winning neuron's learning rate, while c is the bias value. All these parameters decrease toward zero after each training epoch.

Considering digital hardware realization of the NF some other solutions have been proposed, used mostly in the Field Programmable Gate Arrays (FPGA). The solution described in Pena, Vanegas, and Valencia (2006) relies on shifting the bits at each following ring of neighbors to the right. As a result, the value of the term $\eta \cdot G()$ at a given ring is half the value of this term at the preceding ring:

$$G(R, d(i,j)) = \begin{cases} 0.5^{d(i,j)} & \text{if } d(i,j) \le R \\ 0 & \text{if } d(i,j) > R. \end{cases} \tag{5}$$

This concept is very attractive from the hardware implementation point of view, as it does not involve multipliers and dividers and thus is very fast. On the other hand, it significantly reduces the number of values that the NF can take. We have verified this concept by means of the software model of the SOM written in C++, and the results were not satisfactory, especially for large maps. It was not possible to train properly the SOM with more than 20×20 neurons. For this reason we do not consider this solution in the remainder of the paper.

A detailed comparative study for the RNF, GNF and TNF has been presented in Section 3. The simulation results obtained through running the software model of the SOM show that the TNF is a very good approximation of the GNF, while it comes with much less complexity, as only a single multiplication is required in this case. This is discussed in Section 4. Looking only from the software implementation point of view, this conclusion is of minor importance, but in ultra low power devices this is one of the key aspects. Since the majority of existing implementations of the SOMs are software-based, therefore, to best of our knowledge, such comparative investigations have not been carried out so far.

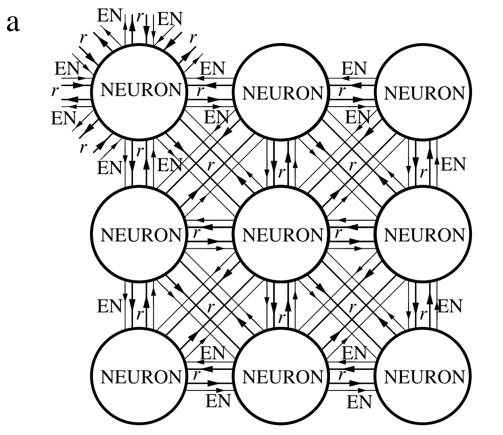
The TF block proposed in this paper is a universal circuit useful in various applications. It offers the steepness of the function that varies in a wide range. The circuit is composed of an asynchronous multiplier and a shifter used to divide the product by selected numbers (powers of 2). In the comparison with a case in which a reciprocal multiplier would be used the number of transistors is reduced approximately by half. If the TF block is used in the SOM (a particular case), the value of the NF is always less than 1, which enables a further optimization. In this case the shifter could be omitted.

One of the important aspects in the design process of the proposed TNF was the minimization of the number of transistors. To achieve this, an additional study has been completed, in which the influence of the signal resolution at the output of the NF block on the learning quality of the SOM has been investigated. The results are presented in Section 4.1. Such investigations are of relevance, as the number of transistors in the NF block i.e. the overall chip area and the energy consumption linearly depend on the signal resolution.

2. Parallel and asynchronous neighborhood mechanism—an overview

The topic of the study is the transistor level implementation of the TF block. For a better illustration, we briefly present some important aspects of the overall neighborhood mechanism. A placement of neurons in the proposed implementation of the Kohonen SOM is schematically shown in Fig. 1(a), with an internal structure of a single neuron presented in Fig. 1(b). Such an arrangement of neurons in the map allows for a very efficient routing between them. Fig. 1(b) presents only these components that are specific for the neighborhood mechanism i.e. the 'enable' signal propagation circuit (EN_PROP) and the radius propagation signal (R_PROP). In this approach, each neuron is connected with only the closest p neighbors, where the value of p depends on the type of the network topology. The proposed mechanism can work with the hexagonal topology (Hex) or rectangular topologies with either 4 or 8 neighbors (Rect4/Rect8), as shown in Fig. 2. The connection between any pair of neighboring neurons requires $2(q + 1)$ signal lines, where q is the number of bits in the signal representing the maximum value of neighborhood range $R_{\max}$ sent in both directions. A single line is required to transfer the enable (EN) signal. The structure of the EN_PROP circuit for an example Rect8 case, as well as the R_PROP block are shown in Fig. 3. The same R_PROP circuit is used in all cases. On the other hand, the EN_PROP circuit for other map topologies differs only in the number of directions. To make the proposed SOM more flexible we have proposed a programmable EN_PROP circuit that enables an easy transition between all the three topologies on a single chip (Długosz & Kolasa, 2008).

The proposed mechanism works as follows: A winning neuron, i.e., the neuron whose 'WSC' identification signal becomes 1 (WSC



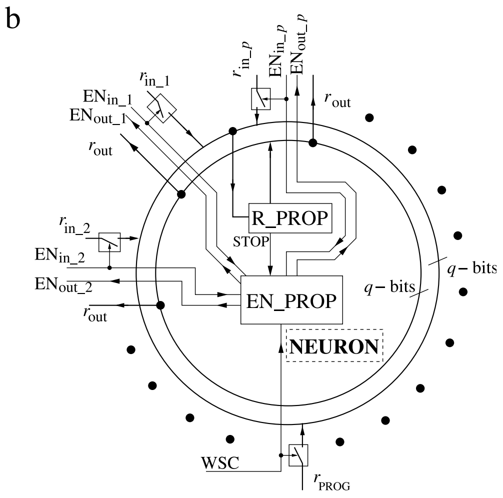


Fig. 1. Diagram of the proposed solution: (a) a placement of neurons in the map, (b) the structure of a single neuron for p neighboring neurons.

signal comes from a Winner Selecting Circuit), is responding by sending a 1-bit EN signal in all directions using the EN_PROP circuit. Particular EN_{out_i} signals of this neuron become the EN_{in} signals in its closest neighbors. The 'WSC' signal is a privileged signal that activates all EN_{out_i} signals. On the other hand, particular EN_{in_i} signals activate only selected EN_{out_i} signals at the opposite side of the EN_PROP block. As a result, the neighboring neurons always receive the EN_{in} signals from only one direction that prevents collisions. Note that the diagonal EN_{in} signals activate three EN_{out} signals, while the horizontal and the vertical EN_{in} signals activate only one EN_{out} signal. Such distinction is necessary, as the number of neurons in each following ring increases, for example by eight for the Rect8 topology. The propagation of the EN signal resembles a wave that spreads asynchronously in all directions concentrically from the winner. The only delay in this process results from a delay caused by a few logic gates located at particular rings.

The EN_PROP block itself does not contain any mechanism that could terminate the propagation of the EN signal at a desired radius R. This problem has been solved by use of the R_PROP circuit, as shown in Fig. 3(b). The R_PROP block uses an additional signal r, where $r = r_{PROG} - d(i, j)$, decreasing its value by 1 at each following ring of neighbors. The propagation of the EN signal terminates at this ring of neurons for which the signal r becomes 0 (STOP=0). Note that the winning neuron receives an r_{PROG} signal that equals the value of the radius R in a given epoch throughout the switch operated by the WSC signal, as shown schematically in Fig. 1(b). The propagation of the r signal is also very fast, as the process

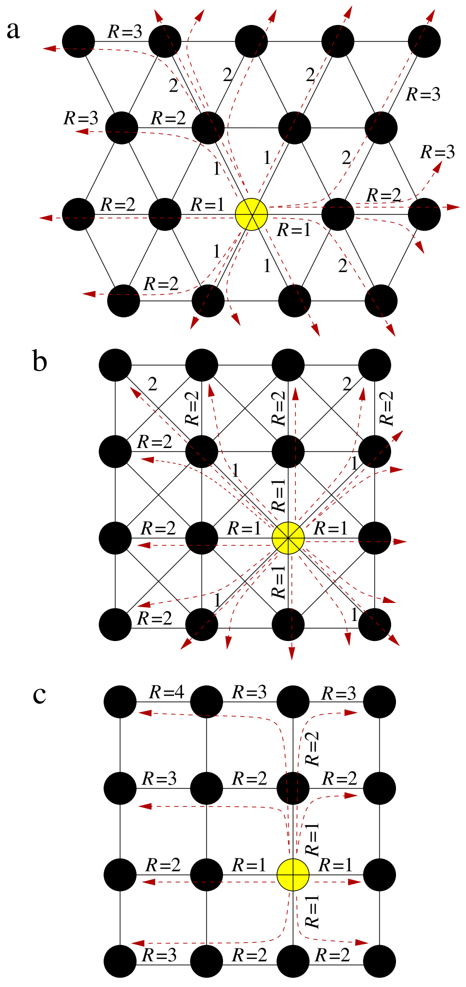


Fig. 2. SOM arranged as: (a) the hexagonal grid (Hex), (b, c) the rectangular grid with 8 and 4 neighbors (Rect8 and Rect4).

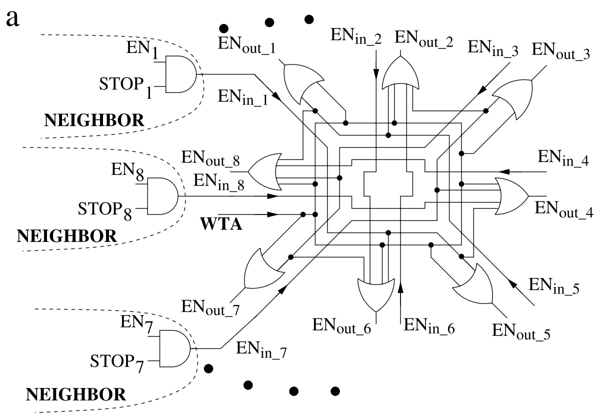
is performed asynchronously. Determining the distances for all neighbors placed at the distance of maximum 15 requires only 11 ns, when implemented in CMOS 0.18 μm technology, and is performed fully in parallel.

The mechanism described above that itself enables the SOM to operate with the RNF, has been extended by a new block. This new circuit operating on the basis of the r signals at particular rings determines the values of the TNF for particular neighboring neurons. Each neuron contains its own NF block and, therefore, all calculations are performed in parallel as well.

3. A comparative study of the quality of the learning process for different neighborhood functions

In this section, a comparative study is presented illustrating how the quality of the learning process depends on the type of the NF and other network parameters. To make the presented results representative the simulations have been performed for different map sizes in the range of 4 × 4 (16 neurons) up to 64 × 64 (4096 neurons). The simulations have been performed for nine general cases i.e. for the three map topologies shown in Fig. 2 vs. the NFs described by (2)–(4). Additionally, we compare the results for two distance measures, namely the Euclidean and the Manhattan ones.

The effectiveness of the learning process of the SOM is evaluated on the basis of five criteria described in Lee and Verleysen (2002). In particular, the vector quantization and the topographic errors



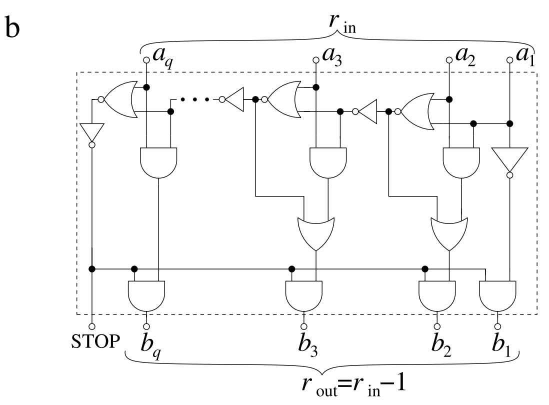


Fig. 3. The EN_PROP block for the Rect8 topology and the R_PROP circuit (the same block is used for all topologies).

are taken into account. The quantization error, which is commonly used to evaluate the learning process of the SOMs is defined as:

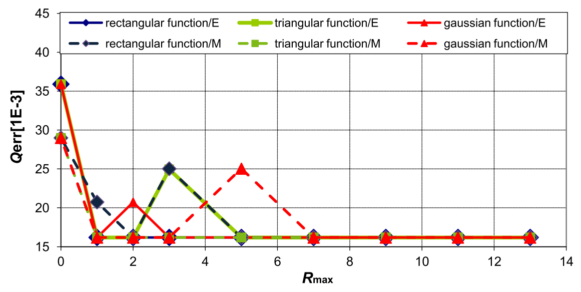
$$Q_{\mathrm{err}} = \frac{1}{m}\sum_{j=1}^{m}\sqrt{\sum_{l=1}^{n}(x_{j,l} - w_{i,l})^2}. \quad (6)$$

In this formula, m denotes a total number of the input patterns X present in the input data set. The quantization error is the error that the NN produces during approximation of the input vector by means of the weight vectors of the winning neurons. This criterion quantifies how the map fits the input data (Uriarte & Martin, 2005). The major disadvantage of this criterion is dependence of the returned values on the number of neurons. For larger maps the value of Q_{err} decreases, as the distances between neurons decrease (Beaton, Valova, & MacLean, 2010). A second measure used to assess the quantization quality is a percentage of dead neurons (PDN), which tells us about the ratio of inactive (dead) neurons versus all neurons. Let us recall that dead neurons are those neurons that never won the competition and as such have not become representatives of any input data. These errors are detrimental to the assessment of the topological order of the map.

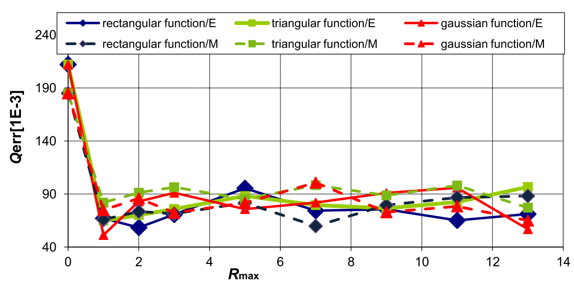
The quality of the topographic mapping is assessed using three measures (Lee & Verleysen, 2002). The first one is the Topographic Error E_{T1}, defined as follows:

$$E_{\mathrm{T1}} = 1 - \frac{1}{m}\sum_{h=1}^{m}\lambda(X_h). \quad (7)$$

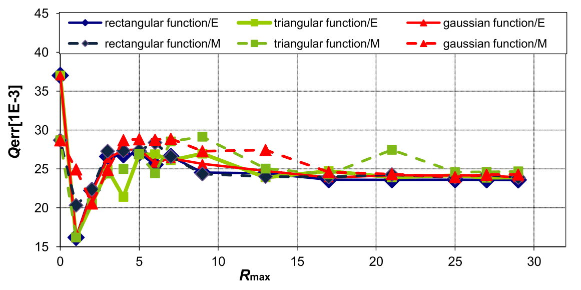
This is one of the errors proposed by Kohonen (Kohonen, 2001; Uriarte & Martin, 2005). The value of $\lambda(X_h)$ is equal to 1 when for a given pattern X two neurons whose weight vectors resemble this pattern to the highest extent are also direct neighbors in the map,



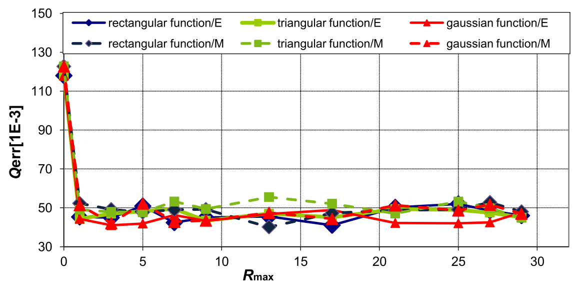
(a) The map with 8 × 8 neurons and 2-D data regularly distributed.



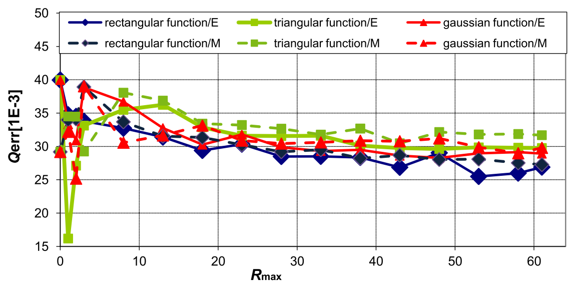
(b) The map with 8 × 8 neurons and 3-D data randomly distributed.



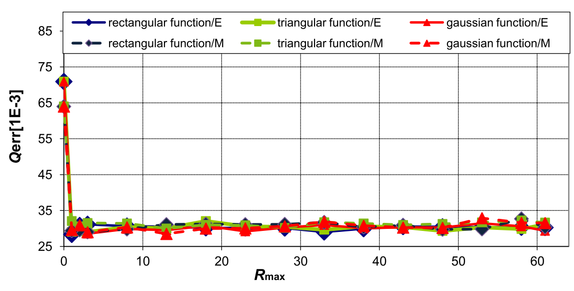
(c) The map with 16 × 16 neurons and 2-D data regularly distributed.



(d) The map with 16 × 16 neurons and 3-D data randomly distributed.



(e) The map with 32 × 32 neurons and 2-D data regularly distributed.



(f) The map with 32 × 32 neurons and 3-D data randomly distributed.

Fig. 4. The quantization error as a function of $R_{\max}$ for particular neighborhood functions, for the Rect4 topology. '*E*' means the Euclidean distance, while '*M*' the Manhattan one.

otherwise $\lambda(X_h) = 0$. The lower is the value of E_{T1}, the better the SOM preserves the topology (Beaton et al., 2010; Uriarte & Martin, 2005). In an ideal case, the optimal value of E_{T1} equals 0.

The remaining two measures of the quality of the topographic mapping do not require the knowledge of the input data. In the second criterion, in the first step, we calculate the Euclidean distances between the weights of an ρth neuron and the weights of all other neurons. In the second step, we check if all p direct neighbors of neuron ρ are also the nearest ones to this neuron in the sense of the Euclidean distance measured in the future space. To express this requirement in a formal manner, let us assume that neuron ρ has $p = |N(\rho)|$ direct neighbors, where p depends on the map topology. Let us also assume that function $g(\rho)$ returns the value that equals the number of the direct neighbors that are also the closest to neuron ρ in the feature space. As a result, the E_{T2} criterion for P neurons can be defined as follows:

$$E_{T2} = \frac{1}{P}\sum_{\rho=1}^{P}\frac{g(\rho)}{|N(\rho)|}. \tag{8}$$

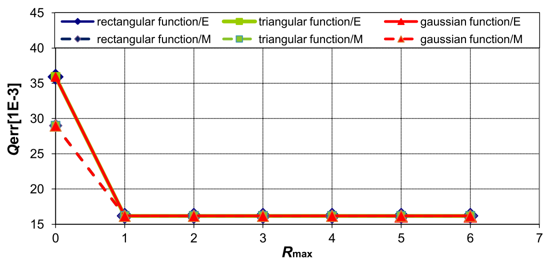
The optimal value of E_{T2} equals 1. Considering the third criterion, we build around each neuron ρ a neighborhood in the feature space (Euclidean neighborhood) defined as a sphere with the radius:

$$R(\rho) = \max_{s\in N(\rho)} \|W_\rho - W_s\| \tag{9}$$

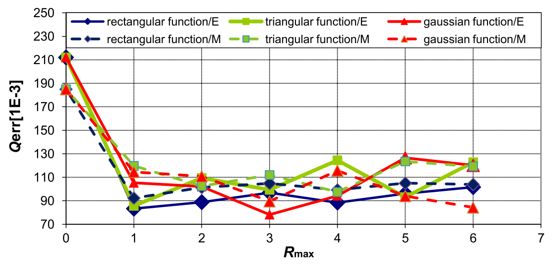
where W_ρ are the weights of a given neurons ρ, while W_S are the weights of its particular direct neighbors. Then we count the neurons, which are not the closest neighbors of the neuron ρ, but are located inside $R(\rho)$. The E_{T3} criterion, with the optimal value equal to 0, is defined as follows:

$$E_{T3} = \frac{1}{P}\sum_{\rho=1}^{P} \left| \left\{ s | s \neq \rho, s \notin N(\rho), \|W_\rho - W_s\| < R(\rho) \right\} \right|. \tag{10}$$

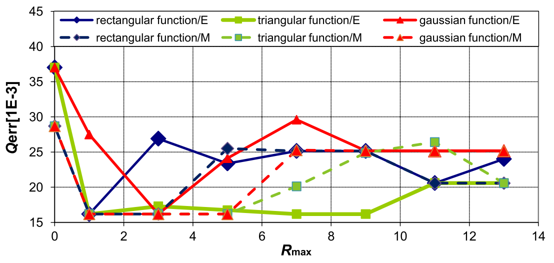
The verification of the SOM has been carried out by running six different learning sets. Five of them were composed of 2-element training patterns (2D), while the 6th one was composed of 3-element patterns (3D). The usage of the 2D sets enables a better illustration of the results (Lee & Verleysen, 2002; Su, Chang, & Chou, 2002; Uriarte & Martin, 2005). Two of the five data sets, as well as the 3D set were divided into B classes (data centers) arranged regularly or randomly in the input data space, where B is the number of neurons in the map. Of particular meaning is the 2D set with the regular arrangement of data centers. Only in this case the learning results can be assessed properly and directly compared for different map parameters (Uriarte & Martin, 2005). In this case the map can achieve the best fit to the input data. To enable a direct comparison of the learning results for different map sizes the input data space is fitted to the number of neurons. For example, for the map with 8 × 8 neurons data are in the range of 0–0.8, for 16 × 16 neurons are in the range of 0–1.6, and so on. As a result, the optimal value of the quantization error (Q_{err}) equals 16.18e−3



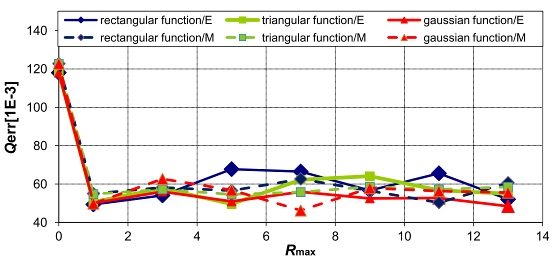
(a) The map with 8 × 8 neurons and 2-D data regularly distributed.



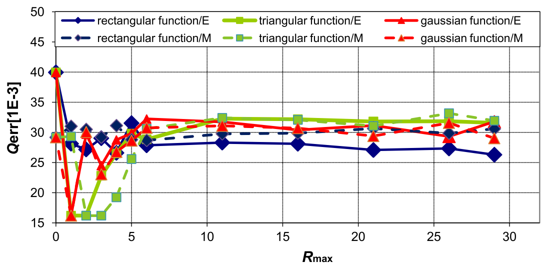
(b) The map with 8 × 8 neurons and 3-D data randomly distributed.



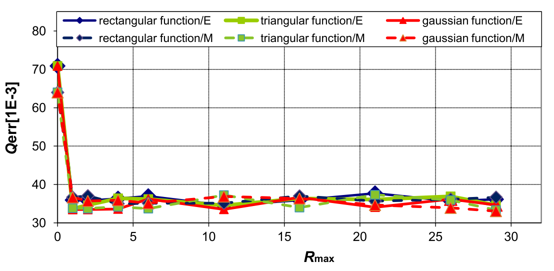
(c) The map with 16 × 16 neurons and 2-D data regularly distributed.



(d) The map with 16 × 16 neurons and 3-D data randomly distributed.



(e) The map with 32 × 32 neurons and 2-D data regularly distributed.



(f) The map with 32 × 32 neurons and 3-D data randomly distributed.

Fig. 5. The quantization error as a function of R_{max} for particular neighborhood functions, for the Rect8 topology. '*E*' means the Euclidean distance, while '*M*' the Manhattan one.

independently on the map sizes, while the optimal values of the PDN/ET1/ET2/ET3 parameters are 0/0/1/0. The optimal nonzero value of Q_{err} results from arrangement of data across particular data centers.

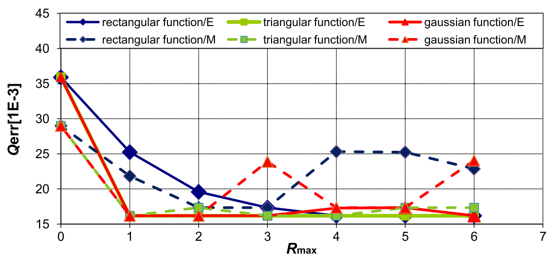
The remaining three data sets were composed of 2000 learning patterns, randomly distributed over the selected regions, as shown in Fig. 8. In these cases input data are in the constant range, independently from the size of the map. As a result, for larger maps the Q_{err} achieves smaller values, as becomes visible in Figs. 4–6(b), (d), (f).

The representative results, for 2D and 3D data placed regularly and randomly in the input data space, are shown in Figs. 4–6 and in Table 1, respectively. These results are presented as a function of the initial value of the neighborhood range, R_{max}, i.e. the value of this range at the beginning of the learning process. The common opinion is that R_{max} should be large enough to cover at least half of the map at the beginning of the learning process. To verify this statement a series of the simulations have been performed for different values of R_{max}. On the basis of the results shown in Figs. 4–6 and Table 1 a conclusion can be drawn that the optimal values of R_{max} are relatively small independently on the map sizes. The values of R_{max} in-between 2 and 7 mean that in particular in large maps the neighborhood range covers only 10%–20% of the map. This conclusion is important as in the hardware realization the value of this parameter exhibits a strong influence on the complexity of the circuit structure. Small values of R_{max} minimize the number of connecting lines between particular pairs of neurons (see Figs. 1(a) and 3(b)) as well as the number of transistors used in the R_PROP block. Smaller values of R_{max} mean also a shorter propagation time between the winning neuron and the most distant neighboring neurons and thus higher data rate.

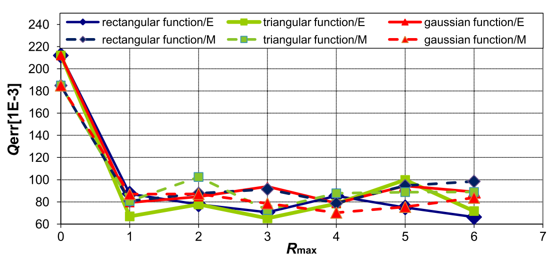
Figs. 4–6 compare the results for the Manhattan as well as the Euclidean distance measures. The results are comparable, which is an important conclusion, as the Manhattan measure does not require the rooting and the squaring operations, and thus it features a simpler hardware structure and much lower power dissipation.

The presented results show that both the TNF and the GNF offer a similar learning quality. This conclusion is very important, as the usage of the TNF significantly simplifies the overall structure of the SOM. Comparing the TNF and the GNF with the RNF one can observe that for small maps up to 8 × 8 neurons all these functions and the map topologies offer a similar learning quality. The situation becomes different in case of large maps. The best results have been achieved for the TNF and the Rect8 topology. Nevertheless, the RNF in many cases offers the optimal results as well, which is important, as in this case the overall mechanism consumes only 20% of energy consumed in case when the TNF is being used. On the other hand, the energy consumed by the TNF is only 10% of energy consumed by the GNF.

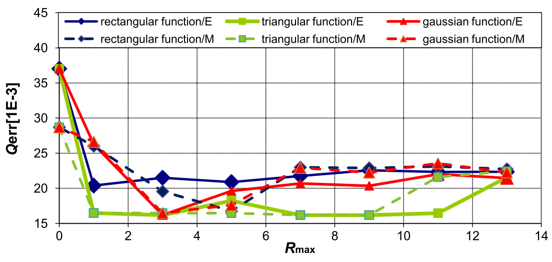
To make the comparison of particular NFs more transparent we performed a test illustrated in Fig. 7. The histograms illustrate the number of cases, for which the map became properly organized (Q_{err} = 16.18e−3) for particular NFs and the topologies, as a function of the map sizes. The largest map for which the optimal values of the five criteria described above have been achieved was the



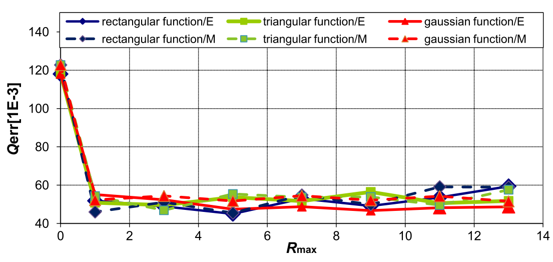
(a) The map with 8 × 8 neurons and 2-D data regularly distributed.



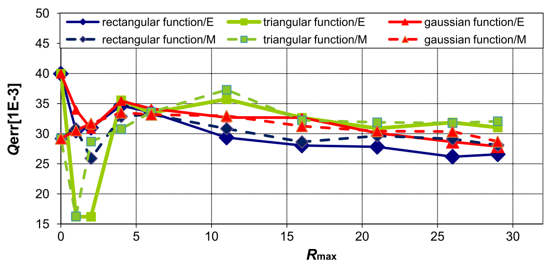
(b) The map with 8 × 8 neurons and 3-D data randomly distributed.



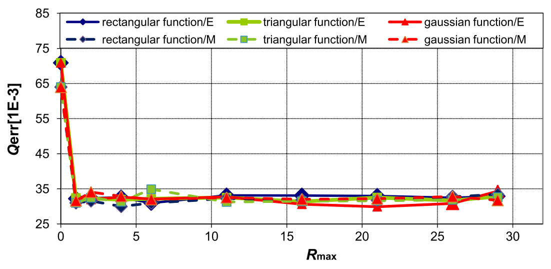
(c) The map with 16 × 16 neurons and 2-D data regularly distributed.



(d) The map with 16 × 16 neurons and 3-D data randomly distributed.

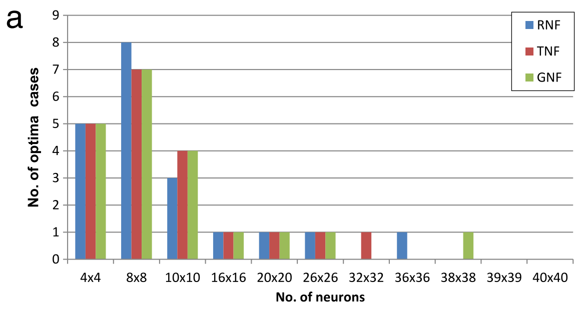


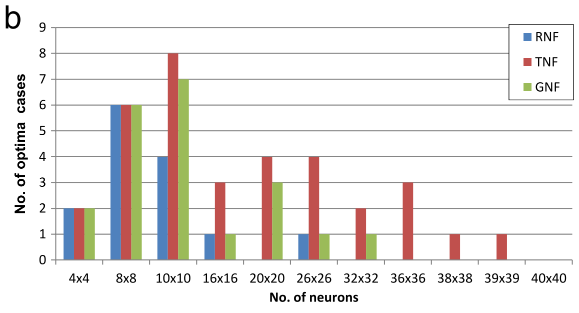
(e) The map with 32 × 32 neurons and 2-D data regularly distributed.



(f) The map with 32 × 32 neurons and 3-D data randomly distributed.

Fig. 6. The quantization error as a function of $R_{\max}$ for particular neighborhood functions, for the Hex topology. '*E*' means the Euclidean distance, while '*M*' the Manhattan one.





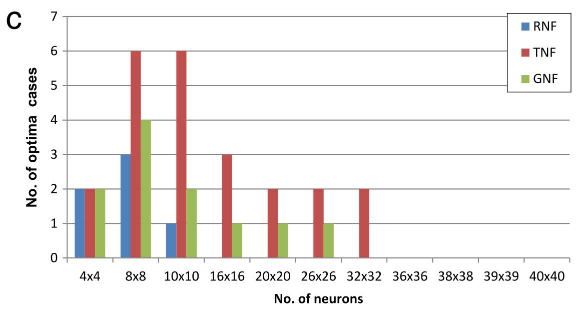


Fig. 7. The number of cases for which the map becomes properly organized, for different map sizes for the following cases: (a) Rect4, (b) Rect8 and (c) Hex topology.

map with 39 × 39 (1521) neurons working with the TNF and the Rect8 topology. Further investigations will be carried out to improve these results. For the Hex topology the results are the worst. The main reasons of this are twofold. On the one hand, the regular input data sets used in these tests favor both the Rect topologies. On the other hand, the rectangular shape of the map causes the

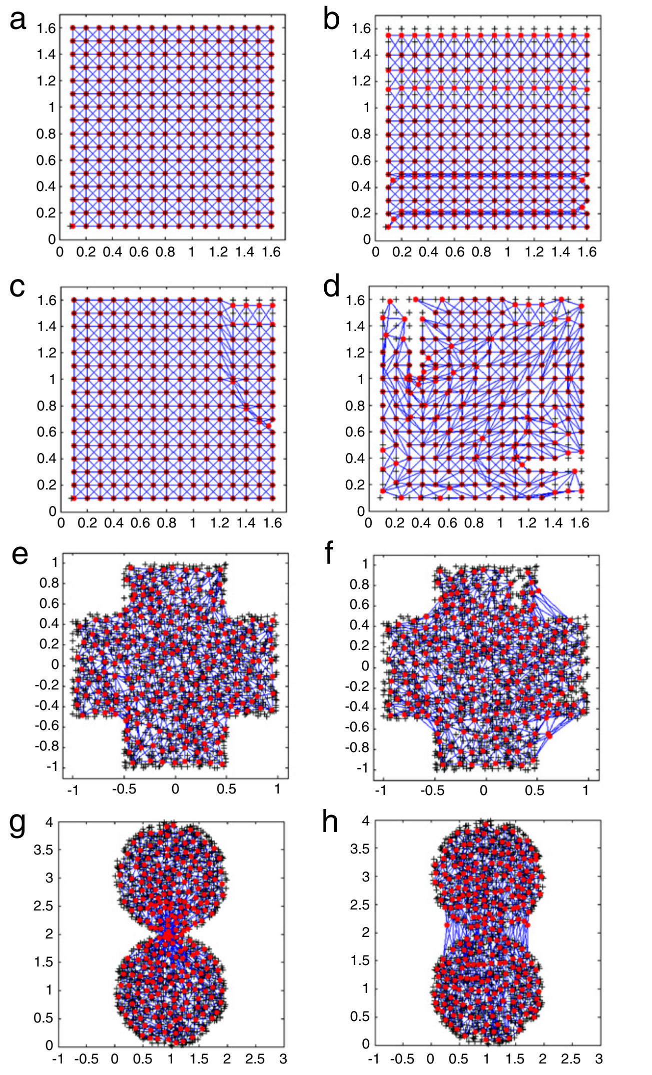


Fig. 8. The quality (Q_{err}/PDN/ET1/ET2/ET3) in the learning process reported for the following cases: (a) RNF/REG/R_{max} = 1 (16.18e−3/0/0/1/0), (b) RNF/REG/R_{max} = 7 (25.1e−3/1.56/0/0.947/0.48), (c) TNF/REG/R_{max} = 3 (17.29e−3/0.391/0/0.989/0.285), (d) RNF/REG/R_{max} = 13 (24.0e−3/2.73/0.066/0.807/5.55), (e) TNF/RAND/R_{max} = 3 (35.46e−3/0/0.004/0.883/2.42), (f) GNF/RAND/R_{max} = 1 (36.06e−3/1.17/0.015/0.852/4.06), (g) TNF/RAND/R_{max} = 1 (53.48e−3/0.781/0.065/0.752/17.27), (h) TNF/RAND/R_{max} = 3 (52.78e−3/1.71/0.049/0.744/6.36).

Hex grid to be irregular, as only part of the hexagon is represented by the square structure of the designed circuit. We observed quite large influence of this effect on the learning quality.

Fig. 8 shows the input data and the final placement of neurons in the input data space for the selected cases coming from

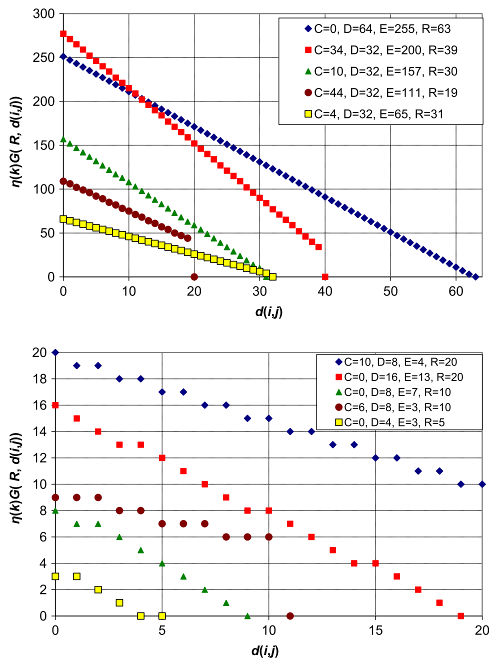


Fig. 10. The shape of the triangular function for selected values of the C, D, E and R parameters. The diagrams illustrate the flexibility of the proposed solution.

Figs. 4–6. The comparison of the results reveals that even small differences in the values of particular criteria (6)–(10) impact the learning quality.

4. The proposed transistor level implementation of the triangular function block

In this section, the hardware implementation of the TF is presented. This block is being used as the TNF block in the proposed SOM. The distances from the winning neuron to particular neurons are determined by r signals at particular rings of neighbors. The winning neuron receives the r_{PROG} signal that determines the neighborhood range R at a given epoch. At each consecutive ring of the neighbors, this signal is decreased by 1, as described above. In case of using the RNF the EN = 1 signal (for any nonzero value of r) determines if a given neuron is in the confines of the neighborhood i.e. is allowed for the adaptation, while the value of the r signal is not important. A different situation is in case of using the TNF, in which the r signals at particular rings of neighbors are used as input signals to the TNF blocks of particular neurons. The TNF block returns the signal that is the $\eta \cdot G()$ term standing in (1). In the next step this signal is being multiplied by the $[X(l) - W_j(l)]$ term, followed by the normalization.

Three parameters are used in this new block. Let us denote them as C, D and E. The proposed TF block realizes the following

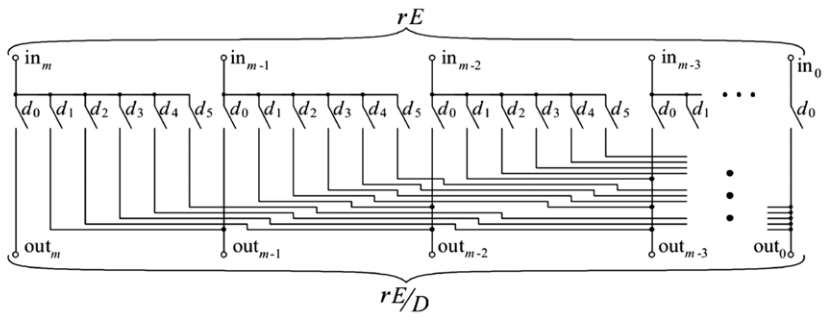


Fig. 9. The proposed circuit and the structure of the bits-shift block, which shifts the bits to the right, thus dividing the signal by D that is always a power of 2.

Table 1
The quality of the learning process treated as a function of the initial neighborhood range R_{max}, for the RNF, GNF and TNF (selected results).

		R_{max}	Q_{err}			PDM			ET1			ET2			ET3		
			RNF	GNF	TNF	RNF	GNF	TNF	RNF	GNF	TNF	RNF	GNF	TNF	RNF	GNF	TNF
Map sizes: 8 × 8																	
REG 2D		0	35.9	35.9	35.9	26.6	26.6	26.6	0.844	0.844	0.844	0.076	0.076	0.076	45.6	45.6	45.6
	Rect4	1	**16.2**	**16.2**	**16.2**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
		3	**16.2**	**16.2**	24.9	**0**	**0**	0	**0**	**0**	0.003	**1**	**1**	0.978	**0**	**0**	0.156
	Rect8	1	**16.2**	**16.2**	**16.2**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
		2	**16.2**	**16.2**	**16.2**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
	Hex	1	25.2	**16.2**	**16.2**	7.81	**0**	**0**	0.02	**0**	**0**	0.975	**1**	**1**	0.766	**0**	**0**
		2	19.6	**16.2**	**16.2**	1.56	**0**	**0**	0.05	**0**	**0**	0.966	**1**	**1**	1.22	**0**	**0**
RAND 3D		0	212	212	212	59.4	59.4	59.4	0.891	0.891	0.891	0.134	0.134	0.134	48.4	48.4	48.4
	Rect4	1	67.0	51.3	64.9	20.3	17.2	20.3	0.338	0.302	0.308	0.598	0.563	0.607	6.42	7.72	6.28
		2	58.4	82.9	70.1	17.2	23.4	20.3	0.259	0.358	0.297	0.670	0.598	0.607	3.64	6.53	5.36
	Rect8	1	83.2	105	85.6	25	29.7	25	0.053	0.073	0.047	0.724	0.767	0.731	7.64	6.94	8.45
		2	88.8	102	109	28.1	31.2	31.2	0.081	0.086	0.091	0.757	0.781	0.731	8.56	6.42	9.42
	Hex	1	87.2	79.4	66.9	28.1	25	21.9	0.173	0.062	0.108	0.792	0.866	0.897	7.30	5.81	3.95
		3	70.6	93.8	65.1	20.3	28.1	20.3	0.156	0.097	0.072	0.811	0.873	0.882	5.62	4.81	3.89
Map sizes: 16 × 16																	
REG 2D		0	37.0	37.0	37.0	28.5	28.5	28.5	0.955	0.955	0.955	0.016	0.016	0.016	196	196	196
	Rect4	1	**16.2**	**16.2**	**16.2**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
		2	22.2	22	20.5	0.391	0	0	0.026	0.043	0.038	0.942	0.947	0.966	0.473	0.41	0.352
	Rect8	1	**16.2**	27.5	**16.2**	**0**	5.08	**0**	**0**	0.052	**0**	**1**	0.81	**1**	**0**	6.11	**0**
		3	26.9	**16.2**	17.3	5.08	**0**	0.39	0.055	**0**	0	0.814	**1**	0.99	5.43	**0**	0.23
	Hex	3	21.5	**16.2**	**16.2**	1.17	**0**	**0**	0.012	**0**	**0**	0.974	**1**	**1**	0.797	**0**	**0**
		7	21.8	20.7	**16.2**	1.17	0.78	**0**	0.014	0.017	**0**	0.976	0.98	**1**	0.977	0.891	**0**
RAND 3D		0	118	118	118	58.6	58.6	58.6	0.966	0.966	0.966	0.031	0.031	0.031	210	210	210
	Rect4	1	45.3	44.3	44.4	25.8	25.8	24.6	0.265	0.263	0.273	0.559	0.543	0.511	9.20	9.38	12.8
		2	45.0	41.0	47.0	24.6	23.0	26.2	0.225	0.258	0.244	0.580	0.565	0.591	6.49	7.32	7.57
	Rect8	1	49.3	50.1	50.6	27.3	28.5	28.5	0.105	0.115	0.112	0.731	0.716	0.683	9.34	9.04	12.6
		5	67.7	50.9	49.5	35.2	28.9	26.2	0.061	0.071	0.095	0.727	0.722	0.728	9.77	11.2	10.1
	Hex	1	51.8	55.1	50.8	29.7	29.7	26.2	0.111	0.161	0.154	0.793	0.809	0.785	7.04	6.92	7.15
		5	45	47.4	53.7	26.2	25	28.5	0.164	0.133	0.143	0.79	0.807	0.806	7.98	7.19	7.38
Map sizes: 32 × 32																	
REG 2D		0	40.0	40.0	40.0	32.3	32.3	32.3	0.990	0.990	0.990	0.010	0.010	0.010	866	866	866
	Rect4	1	34.7	33.3	**16.2**	8.40	6.64	**0.00**	0.208	0.258	**0**	0.701	0.630	**1**	3.83	6.52	**0**
		8	32.7	36.7	35.5	3.52	7.81	6.84	0.141	0.125	0.154	0.775	0.771	0.753	2.63	2.37	2.37
	Rect8	1	28.1	**16.2**	**16.2**	4.79	**0**	**0**	0.009	**0**	**0**	0.878	**1**	**1**	2.71	**0**	**0**
		2	27.2	29.7	**16.2**	2.44	5.86	**0**	0.005	0.157	**0**	0.902	0.634	**1**	1.53	20.1	**0**
	Hex	1	30.5	34.0	**16.2**	4.10	9.57	**0**	0.064	0.123	**0**	0.928	0.780	**1**	2.37	12.9	**0**
		2	31.0	30.8	**16.2**	6.93	5.96	**0**	0.068	0.069	**0**	0.909	0.900	**1**	2.79	3.21	**0**
RAND 3D		0	70.9	70.9	70.9	58.0	58.0	58.0	0.988	0.988	0.988	0.010	0.010	0.010	863	863	863
	Rect4	1	28.4	30.7	29.1	24.8	26.8	25.0	0.306	0.317	0.284	0.501	0.500	0.528	13.27	13.8	10.8
		2	30.8	29.8	29.2	26.6	25.2	25.2	0.279	0.273	0.255	0.548	0.560	0.547	8.40	7.71	9.33
	Rect8	1	35.9	33.5	33.7	31.1	29.4	28.8	0.091	0.101	0.104	0.667	0.693	0.684	14.5	11.5	13.2
		2	35.8	33.5	34.6	31.2	28.8	28.5	0.109	0.102	0.102	0.688	0.700	0.678	11.3	11.5	13.7
	Hex	1	32.2	31.7	32.4	28.7	26.9	27.8	0.168	0.158	0.133	0.755	0.764	0.779	9.30	9.59	8.69
		2	32.4	31.9	32.2	27.9	27.6	27.7	0.133	0.134	0.143	0.787	0.779	0.762	7.79	8.34	10.0

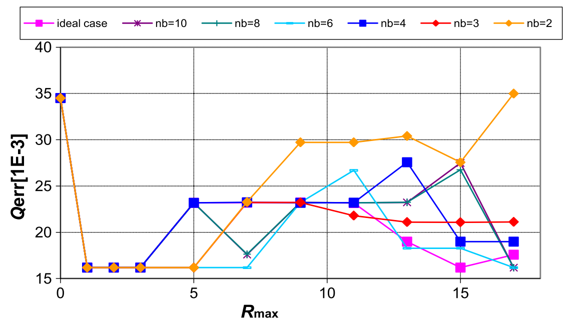
operation:

$$\eta(k) \cdot G(R, d) = r \cdot E/D + C. \tag{11}$$

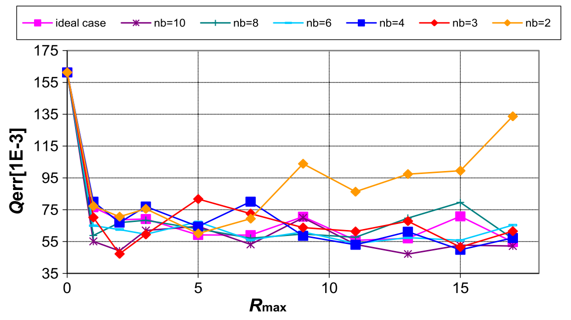
For a given epoch, the values of the C, D, E parameters are equal for all neurons in the map and are reprogrammed only after each epoch. The most complex component is a digital multiplier that is realized as an asynchronous binary tree (BT), utilizing the shift-and-add concept. In the BT concept, at the first layer of the tree the terms corresponding to the bits: 0 and 1, 2 and 3, 4 and 5 and so on, are added in parallel. Then in the next layer the results of the pair 0–1 are added to 2–3, 4–5 to 6–7 and so on. The number of the adders at each following layer is always reduced by half in comparison with the previous layer. In the binary tree approach for a resolution of κ bits a delay, which is introduced by the multiplier, equals $T_{add} \cdot \log_2 \kappa$, where T_{add} is a delay of a single multi-bit adder. To illustrate how does this circuit operate let us consider a simple example in which a binary number 1101 is multiplied by 10011 i.e. we need two layers only. At the first layer we perform two summing operations say, $1 \cdot 100110 + 1 \cdot 10011 = 111001$ and $0 \cdot 100110 + 1 \cdot 10011 = 010011$. At the second layer we shift the first result by two bits to the left and perform the summation with the second result as follows: $11100100 + 0010011 = 11110111$.

Division by D can be performed in various ways. One of the possibilities is the multiplication by the reciprocal for the D variable. In this case the number of bits, nb, at the output of the TF block equals $q + 2 \cdot \kappa$, where q is the width of the r variable, while κ is the width of the E and the D variables. The steepness of the TF varies in the wide range $2^{\kappa} - 1$, $1/2^{\kappa}$, which makes the TF block a universal circuit suitable for various applications. One of the multiplier's inputs can have in this case the width equal to κ bits. For the second input we have to reserve $2 \cdot \kappa$ bits if decimal is at a fixed position and simultaneously the full flexibility is expected. In this case extra zeros have to be added before and after the second variable, resulting in some hardware redundancy in particular multi-bits adders used in BT. To illustrate the problem let us consider an example case of $E/D = 0.11100011$. In this case we multiply the r variable by 00000000.11100011 (8 extra zeros before). If $E/D = 111.00011$ the r variable is multiplied by 00000111.00011000 (5 extra zeros before and 3 after). In this case the output signal width equals 24 while the total number of transistors equals about 4000.

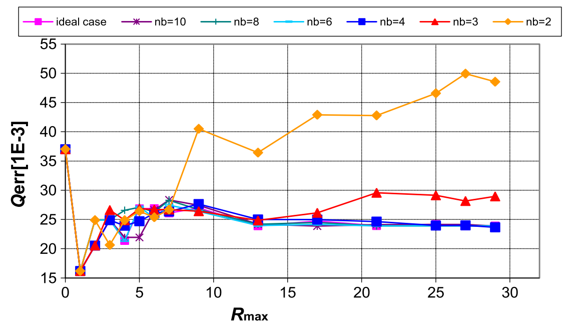
The simulations carried out by means of the software model of the SOM show that the learning process is not affected if the



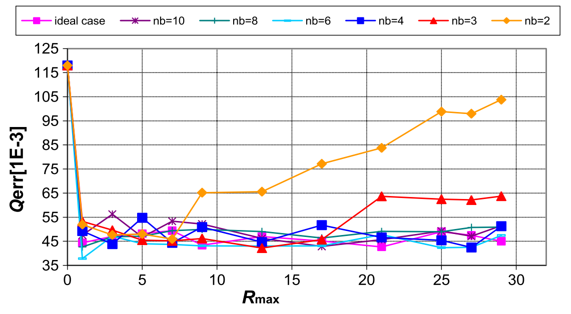
(a) The map with 10 × 10 neurons and 2-D data regularly distributed.



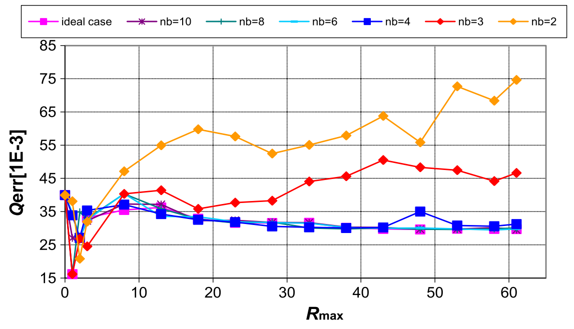
(b) The map with 10 × 10 neurons and 3-D data randomly distributed.



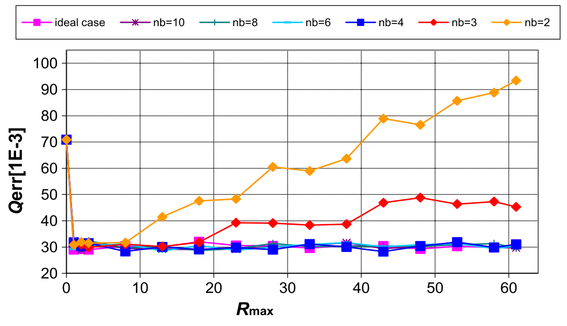
(c) The map with 16 × 16 neurons and 2-D data regularly distributed.



(d) The map with 16 × 16 neurons and 3-D data randomly distributed.



(e) The map with 32 × 32 neurons and 2-D data regularly distributed.



(f) The map with 32 × 32 neurons and 3-D data randomly distributed.

Fig. 11. Quantization error after completing the learning process for different resolutions of the $\eta() \cdot G()$ signal, for the Rect4 topology.

values of the D variable are limited to powers of 2 (i.e. 1, 2, 4, 8, ...). For this reason we propose another solution in which we use the multiplier κ by q bits (ca. 2000 transistors for $q = \kappa = 8$ bits) and an additional divider, shown in Fig. 9. In this case, the division is realized by shifting the bits to the right. The bits-shift operation is performed by the use of a set of switches directly controlled by particular bits of the D variable: $d_0, d_1, d_2, \ldots, d_c$. Only one bit in this variable is allowed to be equal 1, thus the division is based on the following scheme:

$d_0 = 1$ shifts the bits by 0 bits $\rightarrow$ division by 1

$d_1 = 1$ shifts the bits by 1 bits $\rightarrow$ division by 2

$d_2 = 1$ shifts the bits by 2 bits $\rightarrow$ division by 4

...

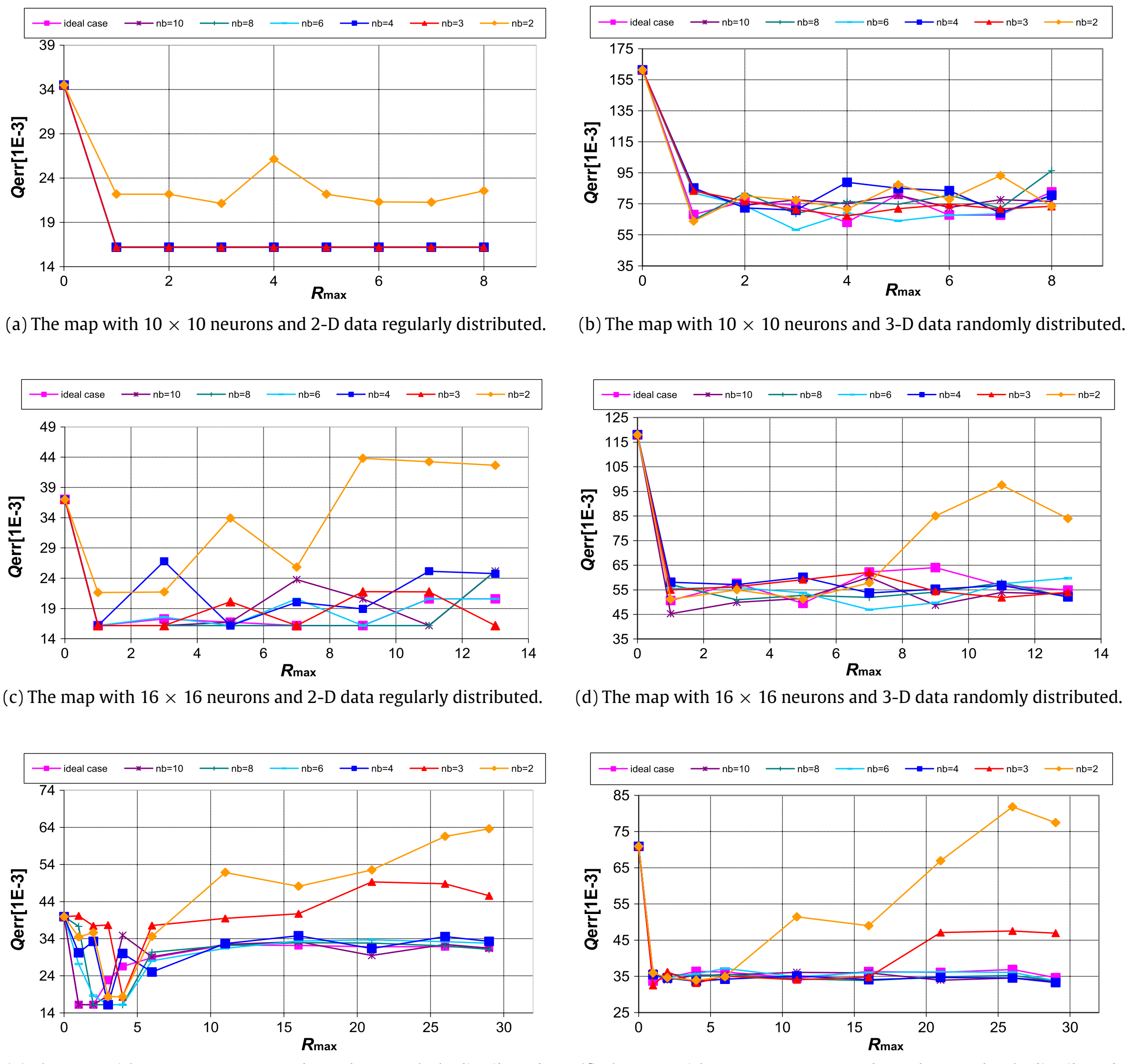
$d_c = 1$ shifts the bits by c bits $\rightarrow$ division by 2^c.

In this approach for an example case of $E/D = 0.11100011$ we multiply the r variable by 11100011 obtaining a 16-bits number and then we shift all the bits by 8 positions to the right. For $E/D = 111.00011$ we also multiply the r variable by 11100011 shifting the output bits by 5 positions. The divider in this case contains ca. 270 transistors (16 bits at the output of the multiplier). In this approach we save half of the chip area but the E and D variables have to be provided separately. On the other hand as in the SOM the E and D variables are reprogrammed seldom (after particular epochs) the problem is rather insignificant.

An additional circuit has to be used in the bits-shift block. Shifting the bits to the right by c positions makes the terminals that correspond to the c most significant bits floating and have to be connected to the ground to avoid the ambiguity at these terminals. This is realized by additional switches (one per each terminal) that are controlled by the signals dependent on particular bits of D. Instead of the switches, realized here as transmission gates (NMOS and PMOS transistors connected in parallel), a series of the AND gates could be used as well, but at the expense of slightly larger number of transistors and larger power dissipation.

Finally, the C parameter is added to the $r \cdot E/D$ term. By a proper selection of the C, D and the E parameters the TNF with any initial value as well as any slope can be realized. To illustrate the principle described above, several triangular functions are shown in Fig. 10 for selected values of these parameters.

The value of the E parameter usually does not exceed 15 or 31 that is sufficient in case of the maps of 16 × 16 to 32 × 32 neurons.



(a) The map with 10 × 10 neurons and 2-D data regularly distributed.
(b) The map with 10 × 10 neurons and 3-D data randomly distributed.
(c) The map with 16 × 16 neurons and 2-D data regularly distributed.
(d) The map with 16 × 16 neurons and 3-D data randomly distributed.
(e) The map with 32 × 32 neurons and 2-D data regularly distributed.
(f) The map with 32 × 32 neurons and 3-D data randomly distributed.

Fig. 12. Quantization error after completing the learning process for different resolutions of the $\eta() \cdot G()$ signal, for the Rect8 topology.

In this case, either 4 or 5 additions are performed in each of the TNF blocks. In SOMs the maximum value of the D parameter should be at least equal to E, since the initial value of the $\eta(k) \cdot G(R, d)$ term theoretically is never greater than 1. As a result, all bits in the $r \cdot E$ product are shifted the most by 5 positions. It must be noted that the number '1' in this case refers to the maximum value that is returned by the function described by (11) and equals $2^{nb} - 1$, where nb is the number of bits at the output of the TNF.

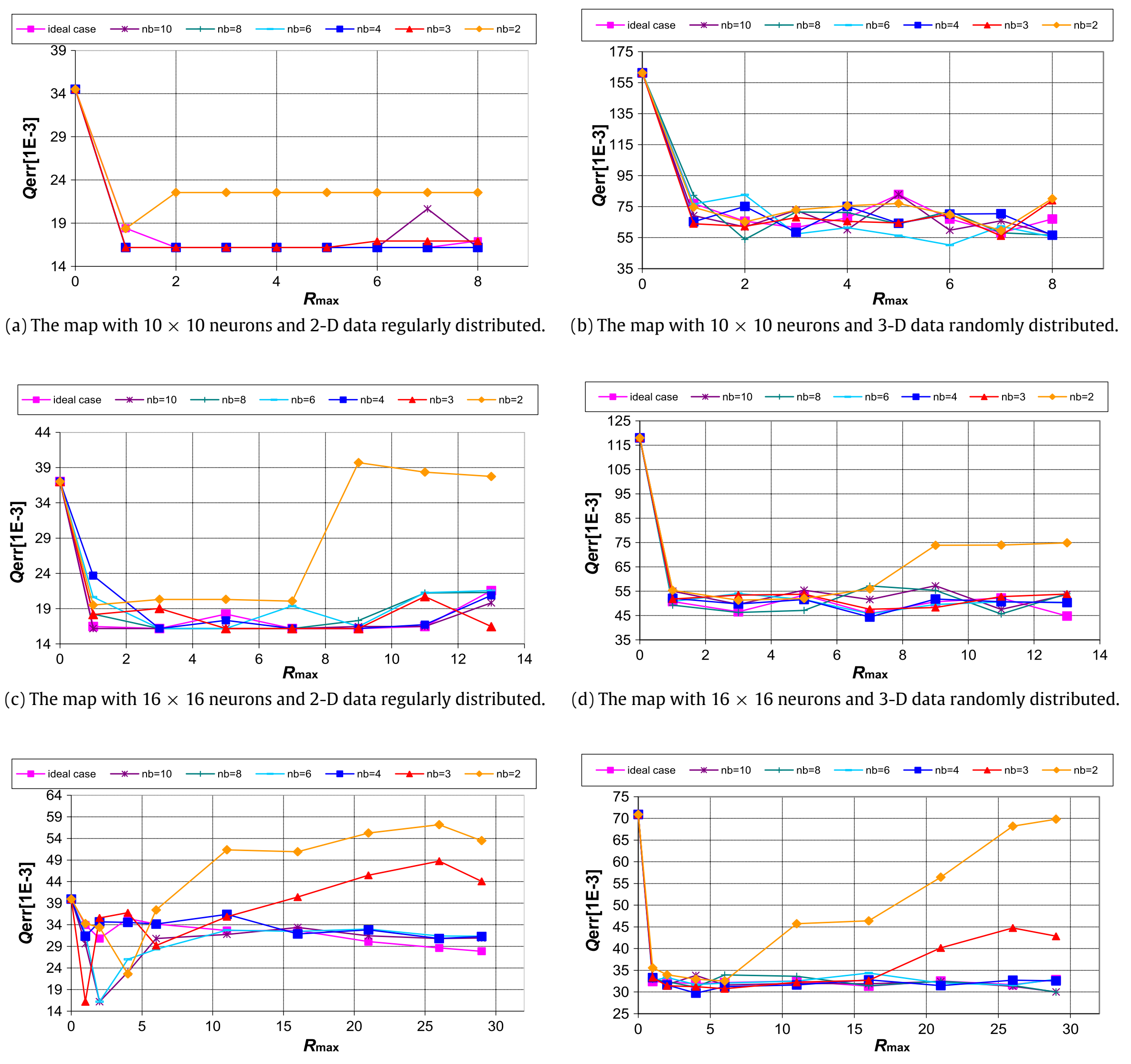
4.1. An influence of the resolution of the TNF output signal on the quality of the learning process

In SOMs realized as digital circuits, the signal resolution at the output of the NF has to be minimized in order to reduce the circuit complexity, as well as the energy consumption. On the other hand, the system level performance of the network cannot be significantly disrupted in this way. In this section, we present selected simulation results of the software model of the SOM for different signal resolutions. The network was trained with data either regularly or randomly distributed in the input data space, as described above. The number of the training patterns was matched for particular map sizes. For example, the map with 16 × 16 neurons was trained with either 1280 or 2560 training patterns which were either regularly or randomly distributed in the input data space, while the map with 10 × 10 neurons was trained with either 500 or 1000 patterns.

The results in Figs. 11–13 again are shown versus the $R_{\max}$, for 10 × 10, 16 × 16 and 32 × 32 neurons in the map, for three network topologies, respectively.

On the basis of the presented results some conclusions can be drawn. In case of regular data it is possible to point out such values of $R_{\max}$, for which the map becomes properly organized for all topologies even for 3 bits of resolution, as shown in Figs. 11–13(a), (c), (e) (exception 11(e)). The Rect8 topology is more robust, as this situation happens for more cases of $R_{\max}$. For smaller maps, e.g., with 10 × 10 neurons, a low resolution does not disrupt the learning process, while for larger maps this effect is visible, as shown in Figs. 11–13(c), (e). Nevertheless, even in this case a proper ordering is achievable for selected values of $R_{\max}$.

A different situation can be observed in the case shown in Figs. 11–13(b), (d), (f). The quantization error varies only moderately. For example, in case shown in Fig. 11(b) the best



(a) The map with 10 × 10 neurons and 2-D data regularly distributed. (b) The map with 10 × 10 neurons and 3-D data randomly distributed.

(c) The map with 16 × 16 neurons and 2-D data regularly distributed. (d) The map with 16 × 16 neurons and 3-D data randomly distributed.

(e) The map with 32 × 32 neurons and 2-D data regularly distributed. (f) The map with 32 × 32 neurons and 3-D data randomly distributed.

Fig. 13. Quantization error after completing the learning process for different resolutions of the $\eta() \cdot G()$ signal, for the Hex topology.

solution has been achieved for the resolution of 6 bits for the Rect4 topology, although for 3 bits a comparable Q_{err} is achievable for selected values of R_{max}. In case shown in Fig. 12(d) for Rect8 topology the best results has been achieved for the resolution of 10 bits, while for 3 bits Q_{err} is 13% larger. Nevertheless, the optimal number of bits in most cases is relatively low, in comparison with typical software realizations.

These conclusions are very important in hardware implementation. Since all neurons in the map are composed of equal blocks, therefore any reduction of the complexity of any block in one neuron has an effect on the complexity of the entire map. The number of transistors in case of the resolution of 3-bits and $R_{max} = 7$ (i.e. also 3 bits) equals c. 550 per a single neuron, while for 6 bits equals c. 1200. In case of an example map with 16 × 16 neurons the number of transistors is reduced by about 170,000. In case of a realization in the CMOS 0.18 μm technology the chip area is smaller by 2 mm^2, while the power dissipation is reduced by 30%.

Transistor level simulations of the proposed TNF block show that, since between the input and the output of this block the number of gates is no greater than 20 (mostly in the multiplier), the propagation time is no greater than 6 ns. As a result, a delay of the overall neighborhood mechanism is comparable for both the RNF and the TNF cases. The problem with using the TNF block is larger number of transistors in this case. In case of the RNF, the number of transistors in the overall neighborhood mechanism per a single neuron equals about 700 for a 5-bits r signal. The new TNF block requires additionally 550–1200 transistors per neuron. This shows that the rectangular NF is much more chip area efficient than the triangular one. On the other hand, it is worth mentioning that since this circuit is digital, the transistor sizes can be the smallest possible for a given technology. As a result, for 5 bits of the resolution the area of a single block equals only c. 7000 μm^2.

The proposed neighborhood mechanism with the proposed TNF is to be used as a component in the analog neural network previously designed by the authors (Długosz, Talaśka et al., 2010). A single neuron in this network with 3 inputs occupied the area of $17,500$ μm^2. For 12 inputs a single neuron would occupy the area of $70,000$ μm^2. This means that the new neighborhood mechanism with the TNF block will increase the chip area of the network by c. 10%–40% (depending on the number of the inputs), which is acceptable in case of the considered maps with 8 × 8 to 40 × 40 neurons.

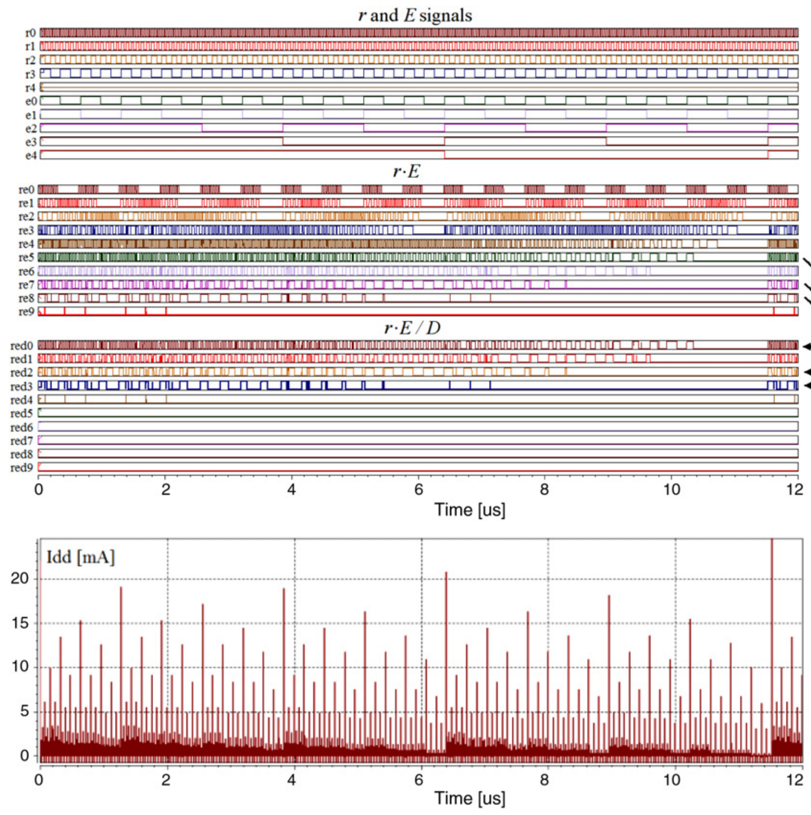


Fig. 14. Transistor level simulations of the TNF block itself. The top plot illustrates different values of the r and the E parameters applied to the inputs. The next two plots illustrate particular stages of the TNF block. The bottom plot illustrates the supply current. The results are presented for the supply voltage $V_{DD} = 1.8$ V.

4.2. Performance analysis of the proposed circuit

In the proposed approach all neurons in the SOM operate in parallel. Transistor level simulations in the CMOS 0.18 μm technology show that independently on the number of neurons the input data rate can be as high as 5–10 MHz, depending on the number of the network inputs and the type of the topology. Each neuron for a single learning pattern X performs about 40 arithmetic operations (for $n = 3$). As a result, the map with 64 neurons performs 25e9 operations/s, at the power dissipation of 20–40 mW. Larger map with 2500 neurons will achieve even 1e12 operations/s. One operation in this case means an addition, multiplication, searching for the winning neuron, etc. For the comparison, in the PC these operations require several clock cycles.

The performance of the TNF block is presented in Fig. 14. In this test, a series of multiplications and divisions is performed for r decreasing from 15 to 0 and E decreasing from 31 to 0 i.e. for 512 combinations. $r \cdot E$ products are then divided by 32 (shifting by 5 bits). Sampling period equals in this case 20 ns but the circuit works properly also for 6 ns. Fig. 14 shows also the supply current, which is proportional to the power dissipation. The height of the current spikes varies in-between 1 and 25 mA, while the width of these spikes is less than 1 ns. As a result, average energy consumption does not exceed a few pJ per a single operation.

It is worth noticing that the shifter consumes a negligible power in the comparison with the preceding multiplier, as the switches in particular branches are reprogrammed very seldom, while the values at the multiplier inputs vary in each learning cycle.

Fig. 15 illustrates the results of the overall neighborhood mechanism for 8 × 8 neurons operating in the Rect4 mode. This mode allows for reaching the highest distances, as only the horizontal and the vertical directions are allowed. Looking from the data rate point of view this is the worst case. The top diagram illustrates the enable signals, EN, in the first column of the map. These signals trigger the adaptation process in particular neurons. Once the EN signal arrives at the bottom row of the map, the propagation starts in this row, as shown in the second plot. A delay between the EN signal at the first (1, 1) and the last (8, 8) neurons in the chain is only 11 ns. Since a delay of a single TNF block equals 6 ns, the entire map is ready for the adaptation after 17 ns. For the Rect8 and the Hex cases, the delay is even shorter as the diagonal directions are also allowed in these cases. The remaining operations performed by the SOM, such as calculation of the distances between a given pattern X and the weight vectors W of particular neurons (the Manhattan measure), detection of the winning neuron (the WSC operation) and the adaptation take another 50–80 ns, depending on the number of the network inputs.

Taking these numbers into account, it can be said that a single neuron in the proposed SOM is able to achieve 40–80 MCUPS (mega connection updates per second). This parameter enables a direct comparison with alternative implementations on FPGA devices. For example, in the FPGA implementation of the SOM described in Hikawa (2005) the network reaches 4.89 MCUPS (4.89e6). Another SOM described in Pena et al. (2006) reaches 28.38 MCUPS. For the comparison, in our proposed transistor level realization the overall SOM with 64 neurons reaches even 2500 MCUPS.

The 3rd and the 4th diagram of Fig. 15 show the supply current, which is proportional to the power dissipation, for the RNF and the TNF respectively. In case of the TNF an average current is approximately 5 times larger. This shows that the RNF function should be used if it enables reaching the optimal learning quality.

An interesting aspect, that is of practical meaning, is the presence of trade-offs between the neighborhood range, the power dissipation, the quantization error, the map sizes, data rate, etc. The value of R_{max} has a dominant impact on the power dissipation, data

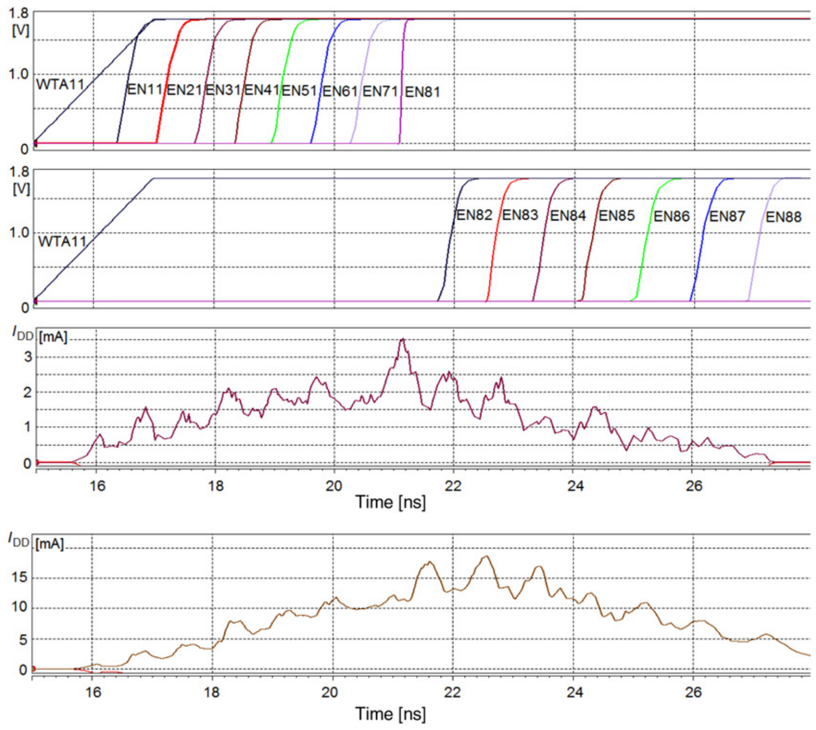


Fig. 15. Transistor level simulations of the overall neighborhood mechanism for 8 × 8 neurons in the map. The results are shown for the Rect4 mode. The 1st and the 2nd plot illustrate the enable, EN, signals at particular stages of the neighborhood. The 3rd and the 4th plot illustrate the supply current for the RNF and TNF cases, respectively.

rate and the chip area. For larger values of $R_{\max}$ larger numbers of neurons are being activated and the R_PROG block becomes more complex. Larger $R_{\max}$ also means a longer propagation time of the EN and the r signals between the winning neuron and the most distant neighboring neurons, resulting in reduced data rate. Fortunately, as has been demonstrated, in the majority of cases the optimal value of $R_{\max}$ is no greater than 7 (3 bits). For this reason, this value can be assumed to be a constant value independent of the map sizes. In this case all neurons have equal structures that simplifies the overall design process of the SOM. The value of Q_{err} has no influence on the power dissipation, as the last parameter depends linearly on the map sizes and data rate only. On the other hand, the Q_{err} does depend on the learning set as well as, to some extent, on the resolution of the signal at the output of the TNF block.

5. Conclusions

A new, flexible, fast and ultra-low power Triangular Neighborhood Function (TNF) block for hardware realized Kohonen SOMs has been developed. The SOM is to be used as a component of ultra low power miniaturized intelligent sensors used in Wireless Sensor Networks in medical applications. In such applications the power dissipation and low chip area are of particular importance. For this reason a series of simulations by means of the software model of the SOM and on the transistor level have been performed in order to optimize the structure of the circuit.

One of the main conclusions is that the TNF can be used instead of the Gaussian one. The learning results are comparable in both cases, while the TNF offers much simpler circuit structure. The simulation results show that in case of both the Manhattan and the Euclidean distance measures the learning quality is comparable, while the Manhattan approach significantly reduces the circuit complexity and the power dissipation.

The other important conclusion is that even low signal resolutions at the output of the TNF block allow for a proper performance of the SOM, while it allows for a further reduction of both the circuit complexity and the power dissipation. In the proposed SOM all neurons operate in parallel. As a result, large neural networks can achieve the computational complexity as high as 1e12 operations/s in the CMOS 0.18 μm technology. In newer technologies (below 90 nm) these numbers will by significantly increased, which will make the proposed solution even more competitive in the comparison with both the PC and the FPGA implementation.

References

Abuelma'ati, M. T., & Shwehneh, A. (2006). A reconfigurable Gaussian/triangular basis function computation circuit. In *IEEE international conference on computer systems and applications* (pp. 232–239).

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, *38*, 393–422.

Beaton, D., Valova, I., & MacLean, D. (2010). CQoCO: a measure for comparative quality of coverage and organization for self-organizing maps. *Neurocomputing*, *73*, 2147–2159.

Bereketli, A., & Akan, O. B. (2009). Communication coverage in wireless passive sensor networks. *IEEE Communications Letters*, *13*, 133–135.

Boniecki, P. (2005). The Kohonen neural network in classification problems solving in agricultural engineering. *Journal of Research and Applications in Agricultural Engineering*, *50*(1), 37–40.

Corbishley, P., & Rodriguez-Villegas, E. (2007). A nanopower bandpass filter for detection of an acoustic signal in a wearable breathing detector. *IEEE Transactions on Biomedical Circuits and Systems*, *1*, 163–171.

Długosz, R., & Kolasa, M. (2008). CMOS, programmable, asynchronous neighborhood mechanism for wtm Kohonen neural network. In *Proc. international conference mixed design of integrated circuits and systems. MIXDES'08. Poland* (pp. 197–201).

Długosz, R., Kolasa, M., & Pedrycz, W. (2010). Programmable triangular neighborhood functions of Kohonen self-organizing maps realized in CMOS technology. In *Proc. European symposium on artificial neural networks. ESANN'10. Belgium* (pp. 529–534).

Długosz, R., Talaśka, T., Pedrycz, W., & Wojtyna, R. (2010). Realization of a conscience mechanism in CMOS implementation of winner takes all neural networks. *IEEE Transactions on Neural Networks*, *21*(6), 961–971.

Dubois, P., Botteron, C., Mitev, V., Menon, C., Farine, P. A., Dainesi, P., et al. (2009). Ad hoc wireless sensor networks for exploration of solar-system bodies. *Acta Astronautica*, *64*, 626–643.

Hikawa, H. (2005). Fpga implementation of self organizing map with digital phase locked loops. *Neural Networks*, *18*, 514–522.

Kohonen, T. (2001). *Self-organizing maps* (3rd ed.) Berlin: Springer.

Kolasa, M., & Długosz, R. (2008). Parallel asynchronous neighborhood mechanism for WTM Kohonen network implemented in cmos technology. In *Proc. European symposium on artificial neural networks. ESANN'08. Bruges, Belgium* (pp. 331–336).

Lee, J. A., & Verleysen, M. (2002). Self-organizing maps with recursive neighborhood adaptation. *Neural Networks*, *15*, 993–1003.

Li, F., Chang, C. H., & Siek, L. (2009). A compact current mode neuron circuit with Gaussian taper learning capability. In *IEEE international symposium on circuits and systems* (pp. 2129–2132).

Macq, D., Verleysen, M., Jespers, P., & Legat, J. D. (1993). Analog implementation of a Kohonen map with on-chip learning. *IEEE Transactions on Neural Networks*, *4*, 456–461.

Masmoudi, D. S., Dieng, A. T., & Masmoudi, M. (2002). A subtreshold mode programmable implementation of the Gaussian function for RBF neural networks applications. In *Proc. IEEE international symposium on intelligent control* (pp. 454–459).

Mokriš, I., & Forgáč, R. (2004). Decreasing the feature space dimension by Kohonen self-organizing maps. In *Proc. 2nd Slovakian–Hungarian joint symposium on applied machine intelligence. Herĺany, Slovakia*.

Peiris, V. (1994). Mixed analog-digital VLSI implementation of a Kohonen neural network. *Ph.D. thesis*. Rozprawa doktorska. Ecole Polytechnique Fédérale de Lausanne. EPFL.

Pena, J., Vanegas, M., & Valencia, A. (2006). Digital hardware architectures of Kohonen's self organizing feature maps with exponential neighboring function. In *Proc. of IEEE international conference on reconfigurable computing and FPGA's* (pp. 1–8).

Su, M.-C., Chang, H.-T., & Chou, C.-H. (2002). A novel measure for quantifying the topology preservation of self-organizing feature maps. *Neural Processing Letters*, *15*, 137–145.

Uriarte, E., & Martin, F. (2005). Topology preservation in SOM. *International Journal of Applied Mathematics and Computer Science*, *1*, 19–22.