ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Programmable Triangular Neighborhood Functions of Kohonen Self-Organizing Maps Realized in CMOS Technology

Rafał Długosz[1], Marta Kolasa[2], and Witold Pedrycz[3]

1-Swiss Federal Institute of Technology in Lausanne, Institute of Microtechnology, Rue A.-L. Breguet 2, CH-2000, Neuchâtel, Switzerland
2-University of Technology and Life Sciences, Institute of Electrical Engineering, ul. Kaliskiego 7, 85-791, Bydgoszcz, Poland
3- University of Alberta, Department of Electrical and Computer Engineering, ECERF Building, Edmonton, T6G 2V4, Canada

Abstract. The paper presents a programmable triangular neighborhood function for application in low power transistor level implemented Kohonen self-organized maps (SOMs). Detailed simulations carried out for the software model of such network show that the triangular function forms a good approximation of the Gaussian function, while being implemented in a much easier way in hardware. The proposed circuit is very flexible and allows for easy adjustments of the slope of the function. It enables the asynchronous and fully parallel operation of all neurons in the network thus making it very fast. The proposed mechanism can be used in custom designed networks either in their analog or digital implementation. Due to the simple structure, the energy consumption per a single input pattern is low (120 pJ in case of the map of 16 x 16 neurons).

1 Introduction

Self-organizing Kohonen neural networks (KNN) belong to the category of networks trained in unsupervised mode [1, 2, 3]. Networks of this type are aimed at the development of dependencies in highly dimensional input spaces. Considering this property they are often referred to as *Self Organizing Maps* (SOMs). KNNs can be realized in various ways. Predominantly we encounter their software implement-tations. Another alternative, whose use is quite limited, deals with a hardware realization in the form of application specific integrated circuits (ASIC). The second approach requires solving specific problems of electronic nature, and is much more challenging. On the other hand, hardware implemented networks offer a fully parallel operation of all neurons in the map and therefore can become much faster than their software counterpart while consuming much less power [4, 5, 6, 7]. This opens quite new possibilities of applications of such networks e.g., in low power portable devices.

The authors of this paper have been working on this type of networks for many years. This paper presents one of the very essential steps in this design process, namely a hardware realization of triangle neighborhood functions.

The results presented in this study show that the triangle function is a very good approximation of the Gaussian function, which due to its complexity is not easy to be realized in hardware. The proposed circuit becomes one of the components of the Winner Takes Most (WTM) KNNs realized as a mixed analog-digital ASIC.

2 Neighborhood functions used in the Kohonen SOMs

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

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

where η is the learning rate, W_j is the weights' vector of a given jth, neuron, and X is the input pattern in the lth presentation. Particular neurons that belong to the winner's neighborhood are adapted with different intensities whose values depend on the neighborhood function $G()$. In the classical approach the rectangular function is used defined as [1, 2]:

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

where $d(i,j)$ is a distance between the winning, ith, neuron and any other, jth, neuron in the map, while R is the neighborhood radius. Much better results are offered by the Gaussian function [3], which is described as follows:

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

The Gaussian function, due to its complexity, is difficult to be realized in large and simultaneously low power SOMs [8]. This was one of the motivations to verify whether the triangular function can be used as a substitute for the Gaussian one. The triangular function which has been used in this case is described as follows:

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

where a is the slope of the function, and η_0 is the value of the winner's learning rate. The parameters a, R and η_0 decrease toward zero after each epoch.

One of the important parameters is the network topology, which is defined as a grid of neurons. This feature determines which neurons belong to the winner's neighborhood for a given value of the radius R [1, 2, 3]. The topologies commonly used are hexagonal as well as the rectangular one with four and eight neighbors. Here we will be referring to them as Hex, Rect4 and Rect8, respectively. Taking the network topology and the neighborhood function into account, we obtain nine different cases, which are investigated for different numbers of neurons in the map.

The effectiveness of the training process in these particular cases described above is evaluated by the quantization error defined as follows:

$$Q_{err} = \frac{\sum_{j=1}^{m}\sqrt{\sum_{l=1}^{n}(x_{j_l} - w_{wl})^2}}{m} \tag{5}$$

where m is the number of learning patterns in the input data set, while n the number of the network inputs.

3 Software simulations for different network parameters

The experiments presented in this section have been performed for the nine cases (3 topologies and 3 functions) mentioned in previous section, for the map sizes varying in-between 4x4 and 32x32 neurons. The comparative results are shown in Figure 1.

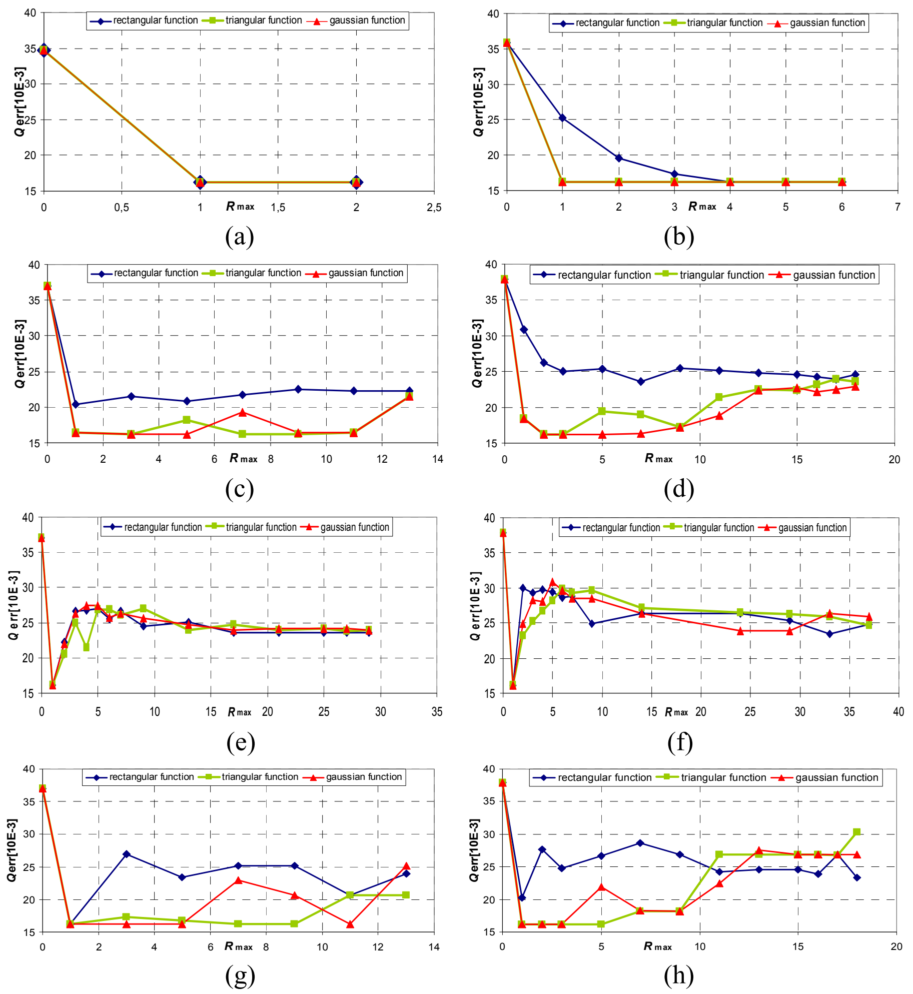

Fig. 1. Quantization error after completing the learning process of the network as a function of the maximum value of the neighborhood radius, $R_{\max}$, for different neighborhood functions, different topologies and different numbers of neurons in the map, for: (a) Hex: 4x4, (b) Hex: 8x8, (c) Hex: 16x16 (d) Hex: 20x20 (e) Rect4: 16x16 (f) Rect4: 20x20 (g) Rect8: 16x16 (h) Rect8: 20x20 neurons.

The results shown in Figure 1 are presented against the varying values of the radius $R_{\max}$, which is the value of the radius R at the beginning of the learning process. The common opinion is that $R_{\max}$ should be high enough to enable the neighborhood range to cover at least half of the map. We have performed a series of the simulations for different values of $R_{\max}$, since from the hardware realization point of view the smallest is the value of this parameter the less complex hardware is required [7].

The presented results show that better results can be obtained even if $R_{\max}$ is very small. For an example case of the map with 20x20 neurons, the optimal values of this parameter are smaller than 4, as shown, for example, in Figure 1 (f).

The presented results are representative for many other experiments performed here. On the one hand, they demonstrate that the triangular function allows for producing very similar results as those obtained in the case of using the Gaussian function and therefore it can be used instead of the last one. On the other hand, the results in case of larger maps are much better than in case of the rectangular function, which justifies using other functions than the rectangular one, although not in all cases. The diagrams (e) and (f) with results obtained for the Rect4 case show that type of the function in this case exhibits almost no influence on the learning process. The similar results have been obtained for smaller maps with 4x4 to 8x8 neurons (not shown for Rect4 and Rect8 cases). In Hex topology the results differ in case of 8x8.

All the considerations above are of limited importance in case of the software realization of the Kohonen SOMs but have a significant relevance, when looking from the hardware implementation point of view, as it will be discussed in next section.

4 Transistor level implementation of the triangular function

We proposed the programmable asynchronous neighborhood mechanism, which is very fast and power efficient [6]. Here, let us recall only the most important aspects. This mechanism generated rectangular functions, with the learning rate η externally programmed as equal for all neurons in the map. The winning neuron sent the radius signal R to its closest neighbors which after decreasing its value by 1 sent it further to the next layer of neurons surrounding the winner, and so on. A propagation of the R signal continues until the R signal reaches zero at a certain layer. The radius signal R was using in this case only to determine the range of the neighborhood. Since this signal propagates simultaneously in all directions of the winner, the entire process is very fast. For example, in a network with 32x32 neurons and realized in the CMOS 0.18μm technology the delay between the winner and the last neuron was about 45 ns.

In this paper, we present an extension of the previous solution, which relies on adding to this mechanism one additional block per each neuron. The radius signal R determines the neighborhood range, as previously, but now it is additionally used as the input signal to this new block that returns the signal that is the $\eta(k)\cdot G()$ term in (1). Three additional parameters are used in this new block. Let us denote them as C, D and E. The proposed triangular function block realizes the following operation:

$$\eta(k)G(i,j,R,d) = R \cdot E/D + C \tag{6}$$

For a given epoch, the C, D, E parameters are the same in all neurons in the map and are reprogrammed after each epoch. The $R\cdot E$ operation is realized using a standard digital shift-and-add multiplier. If in such a block an example binary number 1101 is multiplied by 101 then a series of add operations is performed say, 1·1101+ 0·11010+ +1·110100. Summing the particular terms is realized using typical multi-bit adders. Division is a more difficult operation, unless we limit the allowed values of the D parameter to the numbers that are the following powers of 2 i.e. {1, 2, 4, 8, …}. In this case division is realized by shifting the bits to the right. 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 triangle functions with any initial value as well as any slope can be realized.

To illustrate the principle described above, several triangle functions are shown in Figure 2 for selected values of the C, D, E and the R parameters. The R parameter is the value externally provided to the winner that then decreases at following layers.

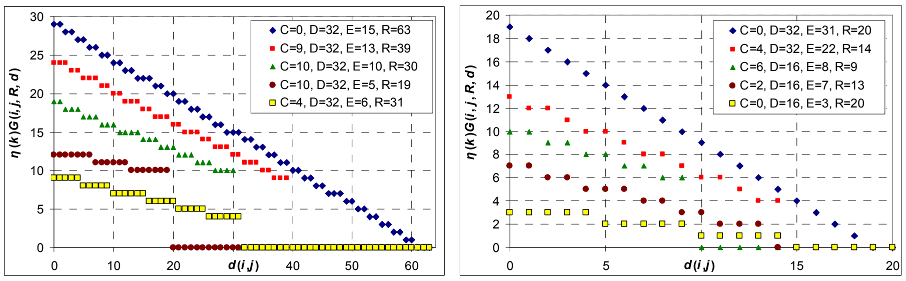

Fig. 2: Parameters of the triangular function for selected values of the *C*, *D, E* and *R* parameters. This illustrates the flexibility of the proposed solution.

The value of the *E* parameter usually does not exceed 15 or 31 that is sufficient in case of maps of 16x16 to 32x32 neurons. In this case, either 4 or 5 additions are performed in each of the triangular function blocks. The maximum value of the *D* parameter should be at least equal to *E*, since the initial value of the $\eta(k)\cdot G(i, j, R, d)$ 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 (6) and equals $2^b - 1$, where *b* is the number of bits. The proposed circuit together with the block that shifts the bits in the product $R\cdot E$ by the number of bits that equals $\log_2 D$ is shown in Figure 3. The values of particular control signals c_{ij}, depend on the value of *D*.

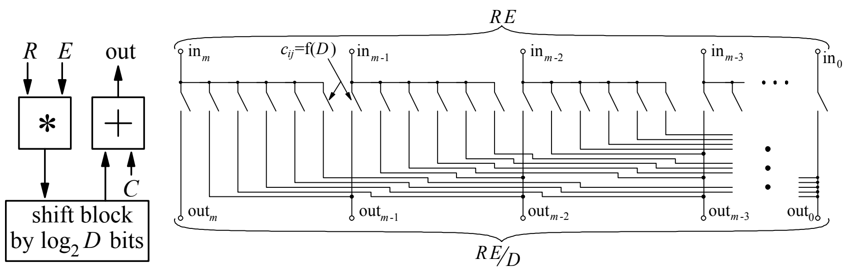

Fig. 3: The proposed circuit and the structure of the bits-shift block, which shifts the bits to the right, thus dividing the signal by *D* that is always a power of 2.

Transistor level simulations of the proposed triangle function 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 several dozen ns that gives comparable results with those obtained by the previous solution, in which only the rectangular function was used. The problem with using this new block is larger number of transistors in this case. In the previous circuit the number of transistors in the neighborhood mechanism per a single neuron was equal to c. 700 for a 5-bits *R* signal. The new block requires additionally c. 1,000 transistors in case of 4-bits the *E* and the *D* parameters or even 1,300 in case of 5-bits. This shows that the rectangular function is much more chip area efficient than the triangular one. On the other hand, it is worth mentioning that since this circuit is a digital solution, the transistor sizes can be the smallest possible for a given technology. As a result, for 5 bits the area of a single block equals c. 7000 μm^2, in the CMOS 0.18μm technology.

The proposed neighborhood mechanism with the proposed triangle function is to be used in the analog neural network previously designed by the authors [5]. A single neuron in this network with 3 inputs occupied the area of 17500 μm^2. For 12 inputs a single neuron would occupy the area of 70000 μm^2. This means that this new block will increase the chip area of the network by c. 10-40%, which is acceptable in case of a medium sized network with 8x8 to 16x16 neurons.

5 Conclusions

A new fast and power efficient triangular neighborhood function for hardware Kohonen SOMs has been proposed and described in the paper. The study presented here can be viewed as one of the steps toward an implementation of fully integrated ultra low power neural network for applications in portable devices that can be used e.g., in the medical diagnostics. The main undertaken efforts were to make the proposed solutions very simple and fast, to reduce the energy consumption as well as the chip area. The results obtained so far show that hardware-realized networks can be much faster than a standard PC computer and much more power efficient [5].

In this paper, we identified two different problems. The first issue was to verify if the triangular function is able to provide similar results as the Gaussian function. Software model results for different network parameters show that both functions lead to a similar accuracy. As a result, we do not need to use the Gaussian function that is much more complex. The other issue was to propose an efficient hardware solution that can approximate the triangular function. The proposed circuit is very fast and needs less than 7ns to calculate the output signal. The used blocks do not require the clock generator, which makes the circuit very simple and power efficient. The energy consumption of a single triangle function block is less than 0.6 pJ per cycle.

References

[1] T. Kohonen, *Self-Organizing Maps*, third ed. Springer , Berlin, 2001

[2] P. Boniecki, "The Kohonen neural network in classification problems solving in agricultural engineering", *Journal of Research and Applications in Agricultural Engineering*, Vol. 50 (1), Poznań, January 2005, pp. 37-40

[3] Mokriš, R. Forgáč, "Decreasing the Feature Space Dimension by Kohonen Self-Organizing Maps", *2nd Slovakian – Hungarian Joint Symposium on Applied Machine Intelligence*, Herľany, Slovakia 2004

[4] D. M. Verleysen, P. Jespers, , and J-D Legat, "Analog Implementation of a Kohonen Map with On-Chip Learning", *IEEE Transactions on Neural Networks*, Vol. 4, No. 3, May 1993, pp. 456-461

[5] R. Długosz, T. Talaska, W. Pedrycz, R. Wojtyna "Realization of a Conscience Mechanism in CMOS Implementation of Winner Takes All Neural Networks", accepted for publication in *IEEE Transactions on Neural Networks*, (2010)

[6] R. Długosz, M. Kolasa, "CMOS, Programmable, Asynchronous Neighborhood Mechanism For WTM Kohonen Neural Network", *15th International Conference Mixed Design of Integrated Circuits and Systems* (MIXDES), Poznań, Poland, June 2008, pp. 197-201

[7] R. Długosz, M. Kolasa, "Optimization of the Neighborhood Mechanism for Hardware Implemented Kohonen Neural Networks", *European Symposium on Artificial Neural Networks* (ESANN), Bruge, Belgium, April 2009, pp. 565-570

[8] Fei Li, Chip-Hong Chang, L. Siek, "A compact current mode neuron circuit with Gaussian taper learning capability", *IEEE International Symposium on Circuits and Systems (ISCAS)*, 24-27 May 2009, pp.2129-2132