Adaptive Weight Change Mechanism for Kohonens's Neural Network Implemented in CMOS 0.18 μm Technology

Tomasz Talaśka[1], Rafał Długosz[2,3,*], Witold Pedrycz[3]

1- University of Technology and Life Sciences, Institute of Telecommunication
Kaliskiego 7, 85-791, Bydgoszcz, Poland
2- University of Neuchâtel, Institute of Microtechnology, Rue A.-L. Breguet 2,
CH-2000, Neuchâtel, Switzerland, *fellow of the Marie Curie OIF fellowship
3- University of Alberta, Department of Electrical and Computer Engineering
ECERF Building, Edmonton, T6G 2V4, Canada

Abstract. In this paper, we present a block of adaptive weight change (AWC) mechanism for analog current-mode Kohonen's Neural Network (KNN) implemented in CMOS 0.18 μm technology. As some other essential building blocks of KNNs dealing with the calculations of the Euclidean distance, formation of a conscience mechanism and a determination of the winner-takes-all (WTA) circuits have been already developed, the AWC forms another essential step towards the realization of the network. We show that the proposed network works with small values of analog signals thus resulting in low power dissipation and chip area when compared with digital realizations of KNNs. Each neuron occupies chip area equal to about 1000 μm^2 and dissipates 20 μW of power for 20 MHz input data rate.

1 Introduction

Continuous progress in microelectronics manifesting in ever increasing packing density has made implementations of large neural networks possible. Such neural networks are required not only to effectively model biological systems but they became a highly versatile optimization vehicle in numerous areas of applications, cf. [1]. One of the key challenges of any effective neural network implementation, which becomes profoundly visible in wireless systems, concerns a low power dissipation of the underlying hardware. Low power dissipation is possible especially in neural networks implemented as analog circuits, where basic operations such as multiplication and summation can be realized in the form of a simple ultra low power circuits. In contrast to digital systems, such analog circuits also occupy lower chip area. In system, where analog KNN is used substantial savings of energy are possible by relocating fast analog-to-digital converters (ADC), which are necessary in digital implementation of KNN to prepare digital input data. Now ADCs can be placed in the system beyond KNN, converting only selected output samples. As a result of such relocation, slower and more power efficient ADCs may be used. In this study, we propose an analog CMOS 0.18 μm implementation of the adaptation weight change mechanism (AWC) for Kohonen's neural networks (KNNs). This mechanism results in one of the main blocks of the KNNs and given our previous realizations, is one of the remaining subcircuits needed toward a final CMOS implementation of the network. Some other another analog blocks of the KNNs architecture have been already successfully implemented by the authors in the standard CMOS AMS 0.8, AMS 0.35 and TMSC 0.18 μm technology [2, 3].

The paper is organized as follows. In Section 2, we discuss an essence of the learning process along with the corresponding ASIC architecture that implements WTA block using Kohonen's rule. In Section 3, presented are details of the hardware implementation of the analog memory being used in the learning as well as simulation of this process. Finally, conclusions are offered in Section 4.

2 The Kohonen network

It is instructive to start with a couple of introductory comments about the paradigm of self-organization (learning) as being supported by Kohonen maps. Typically, the map is organized in a form of a square grid of p x p processing units (neurons) endowed with a vector of connections. The number of inputs (dimension of the input space) is equal to "*n*". The underlying principle of the map is that of a topological preservation of data structure. This means that the map retains the relationships existing in a highly dimensional space when being mapped onto a low, two- or three-dimensional space. As usual, let us assume that at the beginning of learning all weights (connections) are initialized to some random values. The competitive learning process using a winner-take-all (WTA) block relies on presenting the neural network with learning vectors in order to make the weight vectors resemble to presented examples. As a result, only one neuron, the one being the closest to the learning vector (where the closeness is captured through the distance expressed in some metric space) wins the competition. In the sequel only the winning neuron adjusts its connections (weights) in the following way [4]:

$$W(k+1) = W(k) + \eta \cdot (X(k) - W(k)) = W(k) + \Delta W(k) \qquad (1)$$

All other neurons do not change the values of their weights that is:

$$W(k+1) = W(k) \qquad (2)$$

Here $W(k)$ denote a vector of the previous values of neuron's weights, $W(k+1)$ stands for the updated values of the connections; $X(k)$ is a training vector; η is a positive learning rate. The initial value of this rate is in the range (0, 1) and its value remains unchanged or decreases over the progress of the learning , cf. [4].

The proposed KNN's learning module is illustrated in Figure 1. Let us highlight its main components. The Euclidean distance calculation (EDC) implies a degree of similarity between training and weight vectors for each neuron. Given the form of the distance, these calculations are realized using a current squarer [5] being described in the form

$$I_{\text{squar}}(k) = A(x_i(k) - w_{li}(k))^2 \qquad (3)$$

where $w_{li}(k)$ is the *i*-th weight of *l*-th neuron for *k*-th sample, while x_i stands for the *i*-th KNN's input

Here A stands for a certain constant determined by transistor sizing (more specifically, widths and lengths of transistors' channels), x_i and w_{li} are currents that represent inputs and weights of the neurons, while I_{squar} is an squarer's output current. The Euclidean distance in block diagram shown in Figure 1 is represented by current $I_{\mathrm{sq}\,l}$, which for l-th neuron, ($l = 1 \ldots m$, with m being the number of neurons in the system), is defined as:

$$I_{\mathrm{sq}\,l}(k) = -A\sum_{i=1}^{n}\left(x_i(k) - w_{li}(k)\right)^2 \tag{4}$$

The WTA block is used to determine neuron, for which Euclidean distance between training and weight vectors which in our implementation is proportional to I_{sq} attains its lowest value. By doing so we identify the weight vector, which is the closest to the input training vector. Other very useful circuit is the conscience mechanism (CONS). This circuit enables decreasing the number of so called dead neurons. All of described blocks (in various versions) were earlier implemented in analog KNN realized in CMOS 0.18 μm technology [2, 3].

In this paper, we focus on the AWC blocks considering the KNN model presented above. There are various ways to implement a mechanism which enables adaptive changes of neuron's weights. Given the existing alternatives, we have decided to proceed with an analog current-mode approach, due to its simplicity in comparison to both analog voltage mode and digital mode approaches. In current mode the weights are represented by currents and their values are updated by summing up the currents, what can be realized as simple as shorting two signal paths. In voltage mode approach the situation is more complicated. Summing the voltages can be realized using operational amplifiers (OA), which typically dissipate large power.

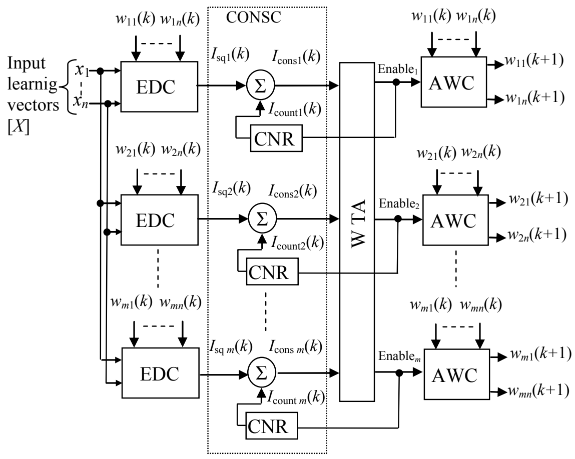

Fig. 1: Diagram of the proposed learning block of Kohonen's neural network for m neurons and n inputs: EDC - Euclidean distance calculation block, CONSC - conscience mechanism with analog counter (CNR), WTA - winner detection circuit, AWC – proposed adaptive (Kohonen's algorithm) weight change mechanism

3 Hardware CMOS realization of adaptive neurons' weight change mechanism

One of the most important elements in hardware implemented AWC blocks are analog memory cells, which store neuron’s weights. Such elements must be accurate in terms of writing and reading. Additionally they must be power and area efficient. Precision has direct influence on convergence of the adaptive algorithm, while power and chip area are important parameters in case of low power CMOS implementation, especially for wireless ultra low power applications.

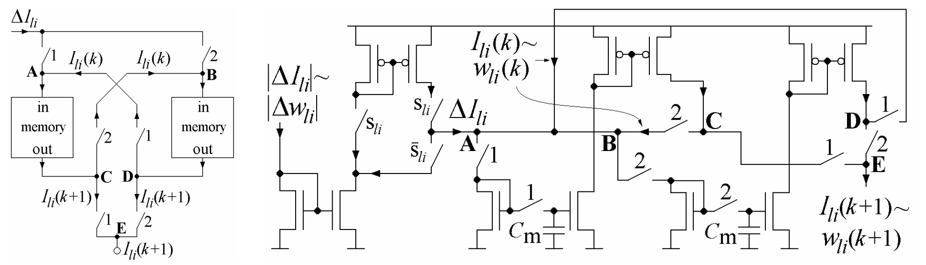

Fig 2: Analog current-mode adaptive weights change mechanism (AWC): (left) general block diagram and (right) electrical scheme.

The proposed AWC block used in our implementation of KNN is shown in Figure 2. This block contains two current-mode sample and hold (S&H) memory elements, with the capacitor memory storages C_m, working alternately. In every time moment one block stores the previous weight $w_{li}(k)$, while the second one is open for accepting new sample $w_{li}(k+1)$, which is the sum of the previous sample and the correction factor $\Delta w_{li}(k)$. Depending on the sign of the difference between signals x_i and w_{li}, factor $\Delta w_{li}(k)$ is added or subtracted from the previous weight sample $w_{li}(k)$. This sign is determined by current comparator, which compares signals x and w, generating output signals s_{li} and $\bar{s}_{li}$, which set up configuration of the AWC block.

Because both neuron weight and the correction factor in the memory block are represented in the form of currents, in the same way we can rewrite (1)

$$I_{li}(k+1) = I_{li}(k) + \Delta I_{li}(k) \qquad (5)$$

There are some advantages and disadvantages of such memory cells. The Simple structure of the cell enables low chip area and leads to low power dissipation, what becomes highly advantageous. One of disadvantages, which are common for many analog memory cells, especially in modern CMOS technologies, is that these memory elements suffer from transistor leakage, thus effectively limiting storage time. The leakage causes that a sample stored in analog memory vanishes with time, introducing some error e. To keep stability of learning process this error must be compensated, what means that factor $\Delta w_{li}(k)$ (only for case, when $x_i - w_{li} > 0$) must be made larger than this error. This factor depends on learning rate η as well as the difference between the input signal x and the weight of the neuron w. For a given value of η, we can identify some minimal value of this difference, which must be preserved to keep the KNN working. This error depends on several circuit's parameters, and can be treated as resolution or error of the KNN implemented in hardware, and expressed as:

$$e = K\frac{t_{\mathrm{U}}}{\eta \cdot C_{\mathrm{mem}}} = K\frac{m}{f_{\mathrm{S}} \cdot \eta \cdot C_{\mathrm{mem}}} \tag{6}$$

In this expression K is the constant parameter, which depends on technology. For m neurons in KNN and sampling frequency f_{S} each neuron is updated on average time of $t_{\mathrm{U}} = m / f_{\mathrm{S}}$ [s]. Increasing f_{S} we decrease t_{U} what minimizes error e. The other way to alleviate this error is by increasing the storage capacitor C_{m}, but this increases the chip area. Both these methods have no influence on adaptive learning process. Increasing the learning rate η also minimizes the error e, but in this case algorithm is affected, especially when learning rate η becomes large.

Concerning practical realization, the proposed KNN has been initially designed and optimized to work at sampling frequencies in the range of 1 - 20 MHz and for applications, where the number of neurons m is less than 30, although is not limited to this value. As a result, depending on sampling frequency, the average value of t_{U} varies in the range between 30 μs and 1.5 μs, respectively. The capacitors used as storage elements have capacitance $C_{\mathrm{m}} = 0.1$ pF. The loss of information in this case is less than 0.03% (assuming the worst case of 30 μs). The learning rate η is set to be in the range 0.1 - 0.4. For $\eta = 0.1$ (the worst case), the leakage is compensated, when input signals x differ from neuron weights w by at least 0.3 % (for $f_{\mathrm{S}} = 1$ MHz).

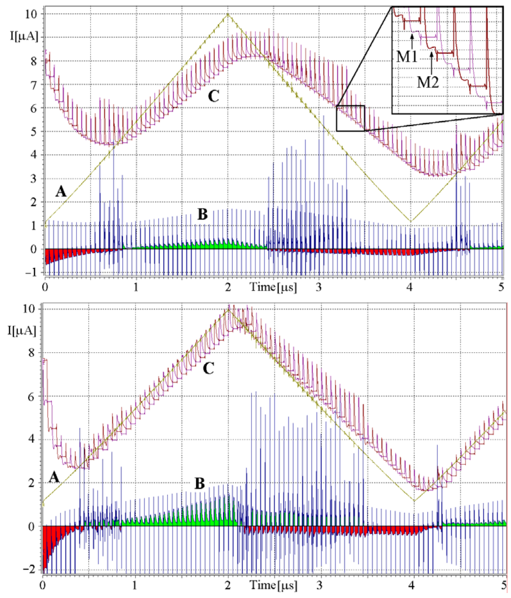

Fig. 3: Experimental result illustrating the adaptation process for the learning rate: (top) $\eta = 0.1$ (bottom) $\eta = 0.4$. A: KNN input signal x; B: $\Delta w = \eta(x - w) \sim \Delta I$; C: the neuron weight w; M1 and M2 signals in both memory cells working alternately.

The example simulation results illustrating the adaptation process for one chosen neuron's weight, when the proposed analog memory cells are used, are shown in Figure 3; they are shown for two different values of learning rate η, i.e. 0.1 and 0.4. In upper Figure alternate operation of both memory cells is also illustrated in zoomed box. In proposed hardware implementation of KNN, each of these signals is represented by current. This is very convenient, because basic mathematical operations such as summing, subtracting are much easier to implement when compared to voltage mode. Summing and subtracting are realized in simple circuit junction, comparing is realized in simple current mode comparators, when only one NOT gate is used, while multiplying by constant coefficient is realized in current mirrors, where transistors have different channel widths.

As could be noted the sign of the Δw (curve B) factor is positive when input signal is larger than neuron weight, and negative in the other case, as expected. In result, the neuron weight w always follows the input signal x. The value of learning rate η directly impacts the speed of adaptation process. While we tested the circuit for a broad range of values of η ranging from 0.05 to 0.8, the best results were obtained for η close to 0.5. For higher values of η adaptation is faster, but the values of the weights oscillate around input signal. As a result, Δw changes its polarity from sample to sample unnecessarily increasing power dissipation of the network, and causes the network may to become unstable.

4 Conclusions

In this study, we were concerned with the CMOS 0.18 µm implementation of analog Kohonen's neural network. One of the most important blocks are analog memory cells, which have direct influence on the adaptation process, and in result on the entire KNN's performance. The proposed cells of analog memory have simple structure, and as result exhibit low power dissipation as well low chip area, what is important in many practical applications of proposed KNN. Considering practical applications of implemented network, the circuit was successfully verified in a wide range of temperatures (namely -20 ÷ +110°C) for different models of transistors. As the overall KNN is still under construction only estimated parameters for KNN may be delivered. For example parameter of KNN with 20 neurons and 8 inputs power dissipation will be 600uW for f_S =20MHz and 30uW for f_S =1MHz.

References

[1] Cauwenberghs G., Bayoumi M.: *Learning on silicon, adaptive VLSI neural systems*, Kluwer Academic Publishers, 1999

[2] Talaśka T., Wojtyna R., Długosz R., Iniewski K.: Implementation of the conscience mechanism for Kohonen's neural network in CMOS 0.18 µm technology, *International Conference Mixed Design of Integrated Circuits and Systems (MIXDES)*, Gdynia, Poland, 2006, pp. 319-315

[3] Długosz R, Talaśka T, Wojtyna R.: New Binary-Tree-Based Winner-Takes-All Circuit for Learning on Silicon Kohonen's Networks, *International Conference On Signals And Electronic Systems (ICSES)*, Łódź, Poland, 2006, pp. 441-446

[4] Kohonen T.: *Self-organizing maps*, Springer Verlag, Berlin, 2001

[5] Bult K., Wallinga H.: A Class of Analog CMOS Circuits Based on the Square-Law Characteristic of an MOS Transistors in Saturation, *IEEE Journal of Solid-State Circuits*, vol. sc-22, No.3, 1987, pp. 357-365