Quasi Analog Formal Neuron and Its Learning Algorithm Hardware

Karen Nazaryan

Division of Microelectronics and Biomedical Devices,
State Engineering University of Armenia,
375009, Terian Str. 105, Yerevan, Armenia
nakar@freenet.am

Abstract. A version of learning algorithm hardware implementation for a new neuron model – quasi analog formal neuron (QAFN) is considered in this paper. Due to the presynaptic interaction of “AND” type, wide functional class (including all Boolean functions) for the QAFN operating is provided based on only one neuron. There exist two main approaches of neurons, neural networks (NN) and their learning algorithm hardware implementations: analog and digital. The QAFN and its learning algorithm hardware are based on those two approaches simultaneously. Weight reprogrammability is realized based on EEPROM technique that is compatible with CMOS technology. The QAFN and its learning algorithm hardware are suitable to implement in VLSI technology.

1 Introduction

An interest in the artificial neural networks and neurons has been greatly increased especially in the last years. Research show, that the computing machines based on NNs can easily solve such difficult problems which require not only standard algorithmic approaches. These neural networks and neurons are based on models of biological neurons [1, 2].

Thus, a biological neuron model is presented in fig.1 (below, the word *neuron* is implied as the artificial model of biological neuron). A neuron, has one or more inputs $x_1, x_2, x_3, \dots, x_n$ and one output y . Inputs are weighted, i.e. input x_j is multiplied by weight w_j . Weighted inputs of the neuron $s_1, s_2, s_3, \dots, s_n$ are summed by resulting algebraic sum S . In general, a neuron has a threshold value, which also is summed algebraically with S . The result of this whole sum is the argument of f activation function of the neuron.

One of the most attractive features of NNs is their capability to be learned and adapted to solve various problems. That adaptability is provided by using learning algorithms, the aim of which is to find the optimum weight sets for the neurons. There exist two main approaches of learning algorithm performance: hardware and software.

In general, neural models differ by the output activation function, input and weight values. For instance, activation function of classical formal neuron (FN) is Boolean one, inputs accept binary and weights - integer values [1, 3]. In analog neuron model, the output function, inputs and weights accept analog values, etc. [4, 5, 6].

Fig. 1. Mathematical model of biological neuron

2 The QAFN Model

Currently, there exist many neural models and variants of their hardware implementations. This work considers a learning algorithm hardware for the new neuron model named QAFN (fig.2) [7, 8], which is based on classical FN and enables the increase of FN functional possibilities.

Fig. 2. Quasi analog formal neuron

On the other hand, research shows [8, 9, 10, 11] that the proposed QAFN model simplifies ways of hardware implementation, with simultaneous improvement of some technical parameters and features: operating speed, accuracy, consuming area in semiconductor crystal, etc.

Thus, QAFN realizes the following mathematical equation:

$$f_{QAFN} = a_1 \sum_{j=0}^{\delta} c_j w_j^+ - a_2 \sum_{j=0}^{\delta} c_j w_j^-, \quad (1)$$

where the activation function f_{QAFN} is a quasi analog one, synaptic c_j inputs accept logic 1 and 0 values, weights w_j^+ and w_j^- accept integer nonnegative (0,+1,+2,+3...) values, a_1 and a_2 are constants, δ is the number of synaptic inputs (fig.3).

The separation of I and II summing parts enable easily to form positive and negative weights, since those parts have identical hardware solutions [9, 10, 11]. Input interaction or presynaptic interaction (PI) unit realizes a logical input interaction between values of vector $\mathbf{X}=(x_1, x_2, \dots x_n)$, by forming logical values of vector $\mathbf{C}=(c_1, c_2, \dots c_{\delta})$. Values of c -s are all possible variants of logical interactions between input values, including all inputs without interactions. Consequently, $\delta=2^n-1$ (where n is the number of information inputs x). For instance, in $n=3$ case, values of c -s are the followings:

$N=3, \delta=7,$	$(c_1, c_2, \dots, c_7) \quad (x_1, x_2, x_3)$
$c_1=x_1$	Without input interaction
$c_2=x_2$	
$c_3=x_3$	
$c_4=x_1 \wedge x_2$ $c_4=x_1 \vee x_2$	Input interaction of logic AND or OR type
$c_5=x_1 \wedge x_3$ $c_5=x_1 \vee x_3$	
$c_6=x_2 \wedge x_3$ $c_6=x_2 \vee x_3$	
$c_7=x_1 \wedge x_2 \wedge x_3$ $c_7=x_1 \vee x_2 \vee x_3$	

Due to PI, the operating wide functional class of QAFN model is provided for both Boolean and quasi analog functions, based only on one neuron (without network construction) [1, 9, 12, 13]. This is one of the most important advantages of QAFN model. The input interaction type ("AND" or "OR") mathematically is not significant for providing QAFN functioning. It should be taken into consideration from the hardware implementation point of view. In this model, the weight values are defined by the weight binary word. Consequently, the weight storing has a simple realization by using EEPROM technique.

From a learning point of view, the neural networks are usually taught easier if the weights of the neurons are allowed to assume both positive and negative values [5]. One of the developed versions of QAFN [9], which provides that opportunity for all weights, is illustrated in fig.3. In this case $a_1=a_2=1$ and only one of the summing parts

(I) or (II) are used, and positive or negative weighted influence is defined by switching the weighted inputs $s_1, s_2, s_3, \dots$ either to positive or negative input of the subtractor:

$$s_j = c_j w_j = c_j (w_j^+ - w_j^-) = \begin{cases} c_j w_j^+ & \text{if } w_j^- = 0; \\ -c_j w_j^- & \text{if } w_j^+ = 0. \end{cases} \quad (2)$$

The most significant bit (MSB) of *weight binary word* defines the sign of the given weight, as it is done in the known computing systems. MSB controls the output of a switch (IV) to the corresponding positive or negative input of the subtractor (III), depending on the weight sign value.

For example, if the number of bits in the weight binary word is eight, the values of w_j are the following:

$w_j^{(7)} w_j^{(6)} \dots w_j^{(0)}$	w_j
0 1 1 1 1 1 1 1	+127
0 1 1 1 1 1 1 0	+126
⋮	⋮
0 0 0 0 0 0 1 0	+2
0 0 0 0 0 0 0 1	+1
0 0 0 0 0 0 0 0	0
1 1 1 1 1 1 1 1	-1
1 1 1 1 1 1 1 0	-2
⋮	⋮
1 0 0 0 0 0 0 1	-127
1 0 0 0 0 0 0 0	-128

Thus, QAFN enables to overcome functional drawbacks of classical neurons, increase functional possibilities and adaptability, and improve technical parameters (speed, features, used area etc.).

3 The QAFN Learning

3.1 The Learning Algorithm

A brief description of a learning algorithm for QAFN will be considered here. The output function $f(k)$ for the k -th combination of input x -s, generating $\mathbf{C}(k) = (c_1(k), c_2(k), \dots, c_\delta(k))$ synaptic inputs, is given by:

$$f(k) = S(k) = \sum_{j=0}^{\delta} w_j(k) \cdot c_j(k) \quad (3)$$

$$W(k_r) = W(k_r + 1) = W(k_r + 2) = \dots = W(k_r + \gamma), \quad (4)$$

where γ is the number of training multitudes [14].

The above described neuron learning algorithm converges after finite number of iterations that is proved in [14].

The brief description of the learning algorithm for QAFN is considered below.

Assigning the output error

$$E = T - A, \quad (5)$$

where T and A are target (desired) and actual outputs, respectively. Considering that T and A accept quasi analog values and x -s, consequently c -s – logic 0 or 1, the learning algorithm for the QAFN, based on the algorithms described in [1, 12, 13, 14], can be written in the following way:

$$W(k+1) = W(k) + \Delta(k) \cdot C(k), \quad (6)$$

where

$$\Delta(k) = \begin{cases} -\varphi & \text{if } E < 0 \text{ (} T(k) < A(k) \text{)} \quad (2\text{nd item}); \\ 0 & \text{if } E = 0 \text{ (} T(k) = A(k) \text{)} \quad (3\text{th item}); \\ +\varphi & \text{if } E > 0 \text{ (} T(k) > A(k) \text{)} \quad (1\text{st item}). \end{cases} \quad (7)$$

Here, φ corresponds to the unit weight value.

It is obvious that E/φ is always equal to an integer value.

Schematically, it can be presented as shown in fig.4. The *output error generator* (OEG) generates the $\Delta(k)$, comparing target (T) and actual (A) values (see equation (7)).

3.2 The Learning Hardware

Computation of synaptic weight values is included within the chip by the digital and analog circuit blocks, and the synaptic weight values are stored in the digital memory.

Before the illustration of the hardware implementation details, a technical particularity should be taken into consideration. The problem is that from the engineering point of view, it is very difficult to provide the exact analog value equal to the target one because of the technological mismatch of the semiconductor components. In order to overcome that problem, the following is recommended to take into consideration:

$$U_{\text{target}} = \frac{U^+ + U^-}{2} \quad (8)$$

$$\Delta U = U^+ - U^- \quad (9)$$

where U^+ , U^- are the bounds of higher and lower voltage levels, between which the actual output value of QAFN is assumed equal to the target one. Of course, ΔU should

be as small as possible considering the hardware implementation properties, in order to reach the ideal case. It is obvious that ΔU and unit weight should have the same order of their values.

Thus, learning target value is given by the analog values of U^+ and U^- considering that the ΔU difference should be the same for all desired output values and input combinations. A voltage level-shifting circuit block (controlled by only U^+) to generate U^+ and U^- values can solve that problem.

The hardware of QAFN learning algorithm is illustrated in fig.5. A simple OEG is designed to generate *up/down* signals. A couple of the voltage comparators generate a digital voltage code comparing the target and actual analog values. That code is then used in the *up/down* signal generation during clock signal appearance. The corresponding digital weight (for which $c_j=1$) is *increased/decreased* by the up/down counter, accordingly. Up/Down signal generation is shown in Table 1. Each value of the weight word bits is stored in a memory cell and is allowed to renew due to EEPROM (electrically erasable-programmable read only memory) technique [15, 16]. To refresh the weights, an access (EN - enable) to the triggers of the corresponding counter, is required. The operational modes of the neuron are presented in Table 2.

Fig. 4. Schematic representation of QAFN learning

Table 1. Up/Down signal generation

Clk	U_{actual}, U_{target}	c_j	up	down	Learning Algorithm
↑	-	0	0	0	item 3
↑	$U_{actual} < U^-$	1	↑	0	item 1
↑	$U_{actual} > U^+$	1	0	↑	item 2
↑	$U^- < U_{actual} < U^+$	1	0	0	item 3

Table 2. The neuron operational modes

Terminal			Weight storage	Counter (cnt.)		Comments
CS	OE	U _{PR}		access to trigger inputs	Trigger outputs	
Working Mode						
1	1	5v	read	-	R _{off}	
Training Mode						
1	1	5v	read	On	R _{off}	Previous weight setting in counter
0	0	5v	R _{off}	Off	cnt.	Learning algorithm realization
0	1	18v pulse	-	Off	cnt.	Previous weight erasing
0	0	18v pulse	write	Off	cnt.	New weight storing

4 Conclusions

The neuron suggested in this paper could be used in complex digital-analog computational systems, in digital filters, data analyzers, controllers, complex Boolean and quasi – analog functioning, digital – analog signal processing systems, etc., where high speed and adaptability are crucial. Weight discretization is the most reliable way of eliminating leakage problems associated with, for instance, capacitive analog storage. Using static digital storage is a convenient solution, it is, however, quite area consuming. Boolean full-functionality and wide functional class for quasi – analog functions are provided by only one QAFN, without network construction, which is one of the most important advantages of the neuron model. A convenient hardware is designed for the neuron learning algorithm performance. The fabrication mismatches of the semiconductor components are taken into consideration, designing a convenient output error generator, based on some engineering approaches. Since the weight correcting value is fixed compatibly with the QAFN hardware and the learning algorithm is performed for only one neuron, the learning rate is relatively high, which depends only on clock signal frequency and possibilities of hardware implementation limitations. The hardware approach of the QAFN learning algorithm implementation provides a substantial high speed for the learning process. Reprogrammability of the weights is realized due to EEPROM technique. The CMOS QAFN and its learning algorithm are suitable to implement in VLSI technology.

Fig. 5. Schematic of QAFN learning algorithm hardware with analog/digital circuit blocks and digital weight storage.

References

1. Mkrtichyan, S.: Computer Logical Devices Design on the Neural Elements. Moscow: Energia (1977), (in Russian).
2. Mkrtichyan, S.: Neurons and Neural Nets. Moscow: Energia, (1971) (in Russian).
3. Mkrtichyan, S., Mkrtichyan, A.S., Lazaryan, A.F., Nazaryan, K.M. and others. Binary Neurotriggers for Digital Neurocomputers. Proc of NEUREL 97, Belgrade, (1997) 34-36.
4. Mead, C.: Analogue VLSI Implementation of Neural Systems. Kluwer, Boston (1989).
5. Haykin, S.: Neural Networks. A Comprehensive Foundation. Macmillan, New York, (1994)
6. Chua, L., and Yang, L.: Cellular Neural Networks: Theory. IEEE Trans. Circuits and Systems, vol 35, N 5, (1988). 1257-1290.
7. Mkrtichyan, S., and Nazaryan, K.: Synthesis of Quasi Analog Formal Neuron. Report of Science Academy of Armenia, N3, Yerevan: (1997). 262-266 (in Armenian).
8. Nazaryan, K.: Research and Design of Formal Neurons (MSc thesis). State Engineering University of Armenia, Yerevan (1997) (in Armenian).
9. Nazaryan, K.: Circuitry Realizations of Neural Networks (PhD thesis). State Engineering University of Armenia, Yerevan (1999) (in Armenian).
10. Nazaryan, K.: Circuit Realization of a Formal Neuron: Quasi Analog Formal Neuron. In Proc of NEUREL 97. Belgrade, (1997) 94-98.

11. Nazaryan, K.,: The Quasi Analog Formal Neuron Implementation Using a Lateral Weight Transistor. Proc of NEURAP 98. Marseilles, (1998). 205-209.
12. Mkrtichyan, S., Lazaryan, A.: Learning Algorithm of Neuron and Its Hardware Implementation. In Proc of NEUREL 97. Belgrade, (1997) 37-42.
13. Mkrtichyan, S., Navasardyan, A., Lazaryan, A., Nazaryan, K. and others.: Adaptive Threshold Element. Patent AR#472, Yerevan, (1997). (in Armenian).
14. Tu, J. and Gonzalez, R.: Pattern Recognition Principles, Mir, Moscow (1978). (in Russian).
15. Holler, M., Tam, S., Castro, H., Benson, R.: An Electrically Trainable Artificial Neural Network (ETANN) with 10240 Floating Gate Synapses. In Proc of IEEE/INNS Int. Joint Conf. of Neural Networks 2, (1989) 191-196.
16. Lee, B., Yang, H., Sheu, B.: Analog Floating-Gate Synapses for General-Purpose VLSI Neural Computation. IEEE Trans. on Circuit and Systems 38, (1991) 654-658.