Mansoura Engineering Journal

Volume 23 | Issue 1 Article 3

3-1-2020

Implementation of Simple Neurons with Complete Hardware-Based Learning Capabilities.

Hassan Soliman
Electronics and Communications Engineering Department., Faculty of Engineering., El-Mansoura University., Mansoura., Egypt., hsoliman@mans.edu.eg

IMPLEMENTATION OF SIMPLE NEURONS WITH COMPLETE HARDWARE-BASED LEARNING CAPABILITIES

" تنفيذ وحدات عصبية بسيطة ذات إمكانية كاملة للتعلم بالدوائر"

by

Dr.Hassan H. Soliman

Comm. &Electronics Dept.

Faculty of Engineering-Mansoura University

ملخص البحث

إن تنفيذ وحدات التشغيل (Neurons) ذات الوصلات القابلة للبرمجة في الدوائر العصبية الصناعية هي مجال مهم للبحوث وتوجه كثير من جهود البحوث لتنفيذ مثل هذه الدوائر باستخدام تكنولوجيا الدوائر ذات التجميع من القياس الكبير جدا (VLSI). ولسوء الحظ فإن مثل هذه التكنولوجيا غير متاحة في كثير من الدول النامية.

في هذا المقال تم تقديم تصميم بسيط لوحدة عصبية بسيطة من نوع ADALINE، هذه الوحدة تتميز بإمكانيات تعلم كاملة منفذة بالدوائر المتاحة تجاريا. وبالرغم من ذلك فإن هذا التصميم يمكن استخدامه وتصنيعه باستخدام تكنولوجيا VLSI .

ومن الجدير بالذكر أن هذا التصميم قد تم تبسيطه بحيث يمكن زيادة عدد أطراف المداخل للوحدة بإضافة مقاومة صغيرة فقط مقابل كل طرف إضافي. وبينما تم تنفيذ الأجزاء الرئيسية من الوحدة باستخدام دوائر تماثلية، مثل مكبرات العمليات، فإن دوائر التحكم في قيمة الوصلات قد تم تنفيذها بدوائر تحكم رقمية.

Accepted February 15, 1998

Keywords

Artificial neural networks, hardware implementations, perceptron-like neurons, ADALINE.

Abstract

In artificial neural networks, implementation of processing units (neurons) which have programmable connection weights is the most process that takes many research efforts. Most of these efforts are dedicated to the implementation using VLSI techniques. Unfortunately, VLSI implementations are not available in most developing countries such as Egypt.

In this paper, simple design for implementing ADALINE-like neurons in artificial neural networks (ANN) with complete learning capabilities is presented. The proposed design is based on the commercially available electronic components, however, it can be easily extended to be implemented using mixed (digital/analog) VLSI technology.

The used components are so minimized to get simple design. Moreover, the number of input connections for the neuron could simply be increased by adding a small resistor for each new connection input. While main forward blocks of the neuron, e.g. summing function, are implemented using analog circuitry, the control of connection weights is implemented using digitally controlled circuitry.

1. INTRODUCTION

Hardware implementation of processing units (neurons) in artificial neural networks (ANN) has attracted great attention. Most of research attentions are directed to VLSI-based implementation due to the complexity, on one hand, and the lack of simple designs that offer complete hardware-based learning capabilities, on the other hand.

The complexity of a design for implementing a neuron, with complete hardware-based learning capabilities, is even raised from the complexity of weight updating circuitry. So, to minimize the hardware required for implementation, the main job is to simplify the weight updating circuitry.

Generally, ANNs can be implemented using analog, digital, optical and/or mixed (analog/digital) circuits. Each type has its own advantages and disadvantages [5,8].

Analog circuits are even used to implement ANNs hardware, such as summers, due to their simplicity compared to digital. However, during learning, the weight updating circuitry, which represents large part of the circuit, are easier to be implemented as digital circuits than in analog circuits.

In this paper, a simple design for a neuron with complete learning hardware-based learning capabilities is presented. The neuron's summing function is implemented as analog circuit, operational amplifier. On the other hand, the weight control is implemented as digitally controlled circuitry.

2. BASIC ADALINE NEURON

Many neural networks are using simple perceptron-like neurons as basic building blocks [9]. This is an adaptive threshold logic element that consists of an adaptive linear combiner cascaded with a hard-limiting element. It is sometimes called Adaptive Linear Element "ADALINE". An adaptive learning algorithm called "Delta rule" is often used to adjust the weights of the ADALINE in order to get the correct response for specific training pattern set during a supervised learning process. Once the ADALINE weights are adjusted, it can be used to classify new unknown patterns. However, a single ADALINE is only capable of classifying or realizing linearly separable logic functions [5,9].

2.1. The Model

The basic model of an ADALINE is illustrated in figure 1. It performs weighted sum of its inputs. The sum is compared to some threshold level. The final output will be turned ON (HIGH) or OFF (LOW) according to whether the threshold level is exceeded or not.

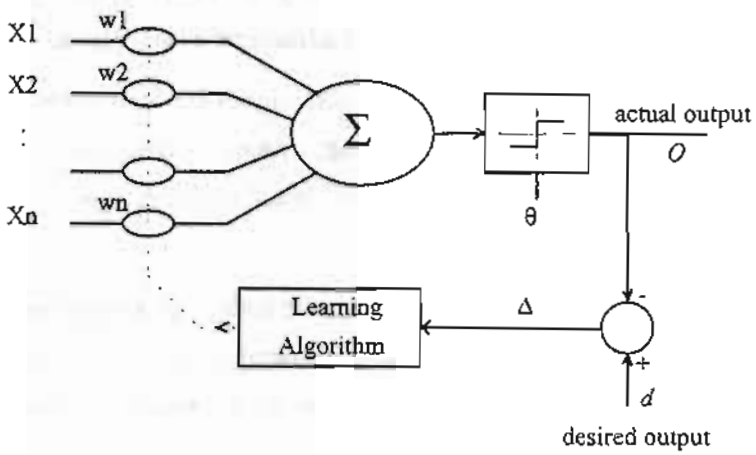

Fig. 1: The basic model of ADALINE element.

To formulate this model mathematically [9], we can write the actual output O as follows

$$O = f\left[\sum_{i=1}^{N} w_i x_i - \theta\right] \tag{1}$$

where: f is the threshold logic function defined as

$$f(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ 0 & \text{if } y < 0 \end{cases} \tag{2}$$

x_i is the i-th input, w_i is the weight of the i-th input, and θ is the threshold bias level.

2.2. The Learning Algorithm

Connection weights and the threshold in an ADALINE can be adapted using different algorithms. Delta rule, originally developed by Rosenblatt [9], is a simple adaptive learning algorithm used to perform the weight adaptation task. This rule may be summarized as follows :

- Connection weights and threshold value are initialized to small non-zero values.
- A new input pattern X with N elements is applied to the input nodes along with the desired output d.

• The actual output is computed through the formula

$$O(t) = f\left[\sum_{i=1}^{N} w_i(t)x_i - \theta\right] \tag{3}$$

• Connection weights are adapted according to the rule

$$w_i(t+1) = w_i(t) + \eta\,\Delta\,x_i \qquad \text{for } 0 \le i \le N \tag{4}$$

where $\Delta = d(t) - O(t)$ and the learning rate coefficient η is a positive number between 0 and 1.

• The process is repeated for all input patterns.

This algorithm includes a gain term η that controls the adaptation rate.

After adjusting the connection weights of the ADALINE, the element will be capable to perform classification of new unknown pattern.

3. PROPOSED IMPLEMENTATION OF ADALINE ELEMENT

In this section, a simple design for implementing ADALINE-like neurons in artificial neural networks (ANN) with complete hardware-based learning capabilities is presented. The proposed design is based on the commercially available electronic components, however, it can be easily extended to be implemented using mixed (digital/analog) VLSI technology.

The used components are so minimized to get simple design. Moreover, number of input connections for the neuron could simply be increased by adding a small resistor for each new connection input. While main forward blocks of the neuron, e.g. summing function, are implemented using analog circuitry, Op. Amp., the control of connection weights is implemented using digitally controlled circuitry. The proposed digitally controlled circuitry is based on binary weighted resistor inserted into the feedback path of the Op. Amp.

The basic idea behind the proposed design may be better explained using the circuit shown in figure 2.

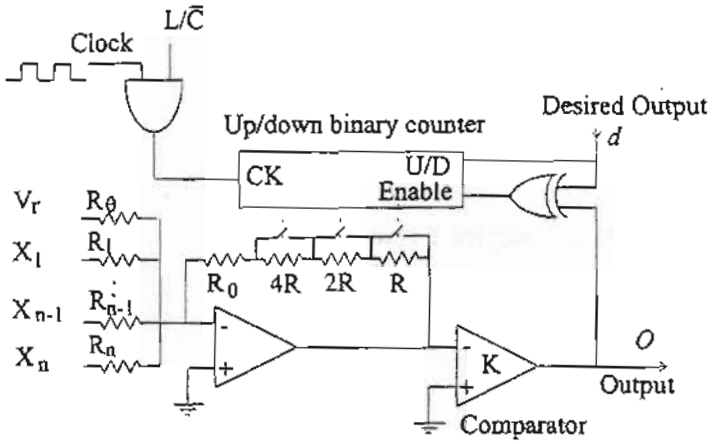

Fig. 2: A Circuit diagram for the proposed ADALINE implementation.

In figure 2, a summing amplifier is used to implement the main function of an ADALINE element. Each input signal, including a threshold voltage V_r, is fed into the amplifier through a small random resistor R_i. The weight of each input signal X_i is determined by the value

$$w_i(t) = -\frac{R_f(t)}{R_i} \tag{5}$$

where $R_f(t)$ is the value of the feedback resistor at the t-th iteration. The value of $R_f(t)$ is determined by the digital control lines fed from the up/down counter used according to the formula

$$R_f(t) = R_o + R\sum_{i=0}^{m-1} B_i \cdot 2^i \quad \Omega \tag{6}$$

where B_i is the i-th bit of counter output, it has either 0 or 1 value.

This gives us variable weights for input signals that can be digitally controlled.

3.1. Operation of the Proposed Circuit

The operation of the proposed circuit is divided into two phases, learning and operation (classification) phase. During the learning phase the input terminal labeled ($L/\overline{C}$) of the AND gate is connected to HIGH state; however, it will be connected to LOW state during the classification phase.

The following procedure will be carried out during the **learning process** :

1. Terminal labeled ($L/\overline{C}$) of the AND gate will set to HIGH to pass the clock signal into the counter to make it ready to count according to the other control terminals, Enable and UP/DOWN.

2. Present an input pattern at the input terminals of the summing amplifier as well as the target output at the desired output terminal of the XOR gate.

3. The output of the amplifier will be determined according to the value of connection weights as

$$y(t) = \sum w_i(t) x_i \tag{7}$$

where $w_i(t) = -\dfrac{R_f(t)}{R_i}$

4. The output of the comparator will be either HIGH if $y(t) \geq 0$ or LOW if $y(t) < 0$.

5. The output of the comparator, i.e. the actual output O, will be compared to the desired output d ; if they are equal the XOR will output LOW, else it will be HIGH.

6. The output of XOR gate controls the counter operation; if it has LOW value, i.e. the desired and actual outputs are equal, the counter will be disabled and its output will be kept constant and the learning phase is finished for that pattern. Else, i.e. XOR has HIGH output, the counter will be enabled to count.

7. The direction of counting will be determined by the value of the desired output. If it is HIGH, i.e. the connection weights must be increased, the counter will count down. In contrast, the counter will be adjusted to count up to decrease the weights if the desired output is LOW. The weight change has the folrmula

$$\Delta w_i = \pm \frac{R}{R_i} \tag{8}$$

4.Steps 3 to 7 will be repeated until the output of XOR goes to LOW state to end the learning phase to that pattern.

5.Steps 2 to 8 will be repeated for all training set patterns.

Table 1 summarizes the operation of the counter according to its control inputs.

Table 1.: The operation of the counter according to its control inputs.

$L/\overline{C}$	XOR output	Desired output d	Counter state	State
HIGH	LOW	LOW	disabled (constant)	***Learning* stop**
HIGH	LOW	HIGH	disabled (constant)	***Learning* stop**
HIGH	HIGH	LOW	counts **UP**	***Learning* running**
HIGH	HIGH	HIGH	counts **DOWN**	***Learning* running**
LOW	X	X	disabled (constant)	***Classification***

During a **classification (operation) process** the following procedure will be carried out :

1.Terminal labeled ($L/\overline{C}$) of the AND gate will set to LOW to prevent the clock signal reaching the counter. This makes the counter output constant, also the connection weights still constant, and no more counting and the other control inputs become irrelevant.

2.Present an input pattern at the input terminals of the summing amplifier.

3.The output of the amplifier will be determined according to the value of connection weights as

$$y = \sum w_i x_i \tag{9}$$

where $w_i = -\frac{R_f}{R_i}$ is the final i-th connection weight after learning.

4. The output of the comparator will be either HIGH if $y(t) \geq 0$ or LOW $y(t) < 0$ to get the final out O.

4. CONCLUSION

A simple design for implementing ADALINE-like neurons in artificial neural networks with complete learning capabilities has been presented. The proposed design can be implemented using the commercially available electronic components and/or using mixed (digital/analog) VLSI technology.

The proposed design permits flexible number of inputs per input pattern, only one small resistor will be added for each input.

A proto-type element has been implemented and tested for realizing different simple logic function such as AND and OR using the available ICs such as CMOS switches (4066), TTL family counters and logic gates as well as operational amplifiers such as LF353.

REFERENCES

[1] H. M. El-Bakry, M. A. Abo-Elsoud, H. H. Soliman, and H. A. El-Mikati, "Design of Neural Networks for Solving Computational Problems", Proceeding of the 13-th national radio science conference, March 19-21, 1996, Cairo, Egypt.

[2] M. A. Abo-Elsoud, H. H. Soliman, H. M. El-Bakry, and H. A. El-Mikati, "A Single Layer Training for High Speed Character Recognition", Proceeding of the 13-th national radio science conference, March 19-21, 1996, Cairo, Egypt.

[3] H. M. El-Bakry, M. A. Abo-Elsoud, H. H. Soliman, and H. A. El-Mikati, "Design and Implementation of Neural Nets for Realizing 2-bit Logic Functions", Proceedings of the 8-th International conference on microelectronics, Dec. 16-18, 1996, Cairo, EGYPT.

[4] F. A. Salam, "Microelectronic implementation of models of neural networks",

IEEE International conference on Circuits and Systems, Dec. 1994, Cairo, EGYPT.

[5] A. Maren, C. Harston and R. Pap, "Handbook of Neural Computing Applications", Academic press, USA, 1990.

[6] B. Mueller and J. Reinhardt, "Neural Networks: An Introduction", Springer-Verlag, Heidelberg, Germany, 1990.

[7] R. Beale and T. Jackson, "Neural Computing: An Introduction", IOP Publishing Ltd., USA, 1990.

[8] C. Mead and M. Ismail, "Analog VLSI implementation of neural systems", Kluwer Academic publishers, USA, 1989.

[9] R. P. Lippman, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp. 4-20, April 1987.

[10] B. Widrow and M. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", Proceeding of IEEE, Vol. 78, No. 9, Sept. 1990.