Modified backpropagation and node perturbation learning algorithms in compact analog VLSI neural networks

Mohamed El-Sayed

Electrical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt.

Learning algorithms and training approaches tailored for compact multilayer feedforward neural networks have been presented. In this type of compact networks, the synapse and neuron building blocks are merged in one unit performing nonlinear weighting of its input. Standard backpropagation and perturbation-based learning algorithms have been modified to use such a type of compact synapse-neuron units. The actual transfer functions of compact unit have been obtained from the simulated hardware characteristics of this unit and then implemented in computer programs based on the modified learning algorithms to perform pseudo chip-in-loop supervised learning. After training, the updated weights are downloaded to the network for feedforward operation. The validity of the modified learning algorithms and training approach has been verified by designing a complete network and training it to perform a function approximation task.

يقدم البحث طرق تعليم واستراتيجيات تدريب مصممة خصيصا للشبكات العصبية التناظرية المدمجة عالية الكثافة. ويتميز هذا النوع من الشبكات بصغر الحجم واستهلاك أقل للطاقة حيث يتم دمج الوحدات الخاصة بالأعصاب والوصلات بين الأعصاب في وحدة واحدة تقوم بعملية وزن لا خطية لدخل الوحدة. وقد تم تعديل وتطويع طريقة الإنتشار الخلفي في التدريب والمعروفة جيدا كأحدى طرق التعليم الموجه في الشبكات العصبية وكذا طريقة الاقلاق لتطبيقها في هذا النوع من الشبكات المدمجة. ولهذا الغرض فقد صممت برمجيات خاصة تعتمد على هذه الطرق وقد روعي فيها أن تكون الدوال الرياضية المعبرة عن الوحدات المدمجة مطابقة تماما لنظيراتها المصممة باستخدام تقنيات الدوائر المتكاملة عالية الكثافة. ولاختبار طرق التعليم واستراتيجيات التدريب المقترحة فقد تم عمل تصميم ومحاكاة لشبكة عصبية مدمجة كاملة وتدريبها بنجاح في مجال تقريب الدوال.

Keywords: Analog VLSI, Compact neural network, Back-propagation, Node-perturbation.

1. Introduction

In Artificial Neural Networks (ANNs), standard packpropagation [1], the most popular gradient descent learning algorithm, assumes that the synapses are linear multipliers and the neurons have sigmoidal nonlinearities. Although it is possible to design analog circuits that approximate these characteristics [2-4], the results is rather large synapses and neurons and thus, an expensive solution. A better approach is to use an algorithm that is more tolerant of analog nonidealities. For example, if an algorithm does not assume that the synapse is a linear multiplier and the neuron has sigmoidal nonlinearity, much more compact synapse and neuron units can be used. The optimum solution is to merge both the synapse and neuron in one unit that performs a nonlinear weighting for its input.

Standard back-propagation learning algorithm can be modified to take into account such a type of nonlinear multipliers. This approach has been applied in the Kakadu neural network chip [5], which is trained using chip-in-loop supervised learning. In this training approach, the chip is used in the forward pass and a host computer is used in the feedback pass and makes use of neuron outputs to compute the local error gradients and then updates the synaptic weights. The disadvantage of this approach is the huge wiring requirements, since all neurons of the network have to be accessible.

Perturbation-based on-chip gradient descent learning algorithms are also well suited to analog VLSI implementations because they do not make assumptions about

synapse and neuron circuit characteristics. The simplest perturbation algorithm is called Serial Weight Perturbation (SWP) [6], in which the weights are perturbed in sequence and the change in the error resulting from the perturbation is measured. This algorithm is extremely tolerant of nonideal analog circuits and thus, extremely compact circuits can be used. Unfortunately, SWP is very slow especially in the case of fully connected large networks. The Chain Rule Perturbation (CHRP) algorithm [7,8] provides the tolerance of nonidealities of SWP along with a significant speedup. In this algorithm, the weights are updated using composite perturbations of hidden neuron outputs and weights. Although this method is very time efficient compared to SWP, it requires extra wiring to perturb all hidden nodes within the network.

Recently, a pseudo chip-in-loop training approach based on standard back-propagation learning algorithm has been proposed [4]. In this approach, the transfer functions representing the multiplication process (synapse function) and the sigmoidal nonlinearity (neuron function) are obtained from the simulated hardware versions and then fitted by mathematical equations and implemented in a computer program for off-chip learning. After training, the final weights are downloaded to the network for feedforward operation. This approach can be extended to the case of modified backpropagation and perturbation learning algorithms, resulting in a reliable training approach in compact VLSI neural networks where nonlinear synapses are in use.

In the present paper, modified backpropagation and perturbation-based (called Node Perturbation (NP)) learning algorithms with nonlinear synapses are proposed. A complete design aided with SPICE simulations of compact synapse-neuron unit is also presented. The validation and verification of the proposed learning algorithms and training approach have been carried out through design and simulation of a compact analog Multi-Layer Feed-Forward Neural Network (MLFFNN) trained as a function approximator, using pseudo chip-in-loop training approach.

2. Modified backpropagation learning algorithm with nonlinear synapses

In the gradient descent algorithm [9], to which the standard backpropagation learning is belonging, a synaptic weight is updated opposite to the direction of the gradient of the error signal E in the weight space. The error signal E(n), at the n'th iteration, is normally taken as the instantaneous sum of squared errors overall the output neurons. Thus, E(n) can be expressed as

$$E(n) = \frac{1}{2}\sum_{j=1}^{N}[d_j - y_j(n)]^2 , \qquad (1)$$

where y_j and d_j are the actual and desired response of the output neuron j, respectively, and N is the number of neurons in the output layer. If P denotes the total number of patterns contained in the training set, the average squared error E_{av} is given by

$$E_{av}(n) = \frac{1}{P}\sum_{p=1}^{P} E_p(n) . \qquad (2)$$

The objective of the learning process is to adjust the free parameters of the network (i.e, synaptic weights and thresholds) so as to minimize E_{av}. A synaptic weight $w_{ji}(n)$, connecting a neuron i in a given layer to a neuron j in the next layer, is updated with an increment $\Delta w_{ji}(n)$, which is given by

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} , \qquad (3)$$

where η is a proportionality constant determining the learning rate. The partial derivative in the right hand side of Eq. (3) can in general be expressed as

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial w_{ji}(n)} . \qquad (4)$$

The quantity $\partial E(n)/\partial y_j(n)$ is normally known as the local error gradient $\delta_j(n)$ of the neuron j.

Depending on whether the neuron j is located in the output layer or in a hidden layer, the two partial derivatives in the right hand side of Eq. (4) can be evaluated as follows: Consider first that the neuron j is an output neuron. In this case, $\delta_j(n)$ is obtained directly from Eq. (1) as

$$\delta_j(n) = \frac{\partial(E_n)}{\partial y_j(n)} = -[d_j - y_j(n)] = -e_j(n), \qquad (5)$$

where e_j is the error signal. In the conventional neuron model used in MLFFNNs, the actual neuron response y_j is simply obtained by weighting the input signals applied to this neuron, summing all weighted inputs and then activating this sum according to a certain neuron nonlinearity (e.g. sigmoidal nonlinearity). This model considers that the synapse unit is an ideal multiplier and the nonlinearity is applied at the neuron level. In the more compact neuron model, in which the nonlinearity is applied at the synapse level, nonlinear multipliers can be employed to model the synapse units, and the function of the neuron unit is simply to sum the synapse outputs connected to it. Thus, $y_j(n)$ can be expressed as

$$y_j(n) = \sum_{i=0}^{M} f[w_{ji}(n), x_i(n)], \qquad (6)$$

where f is the nonlinear synapse-neuron transfer function, x_i is the i-th input signal and M is the number of the input signals connected to the neuron j. Note that for mathematical convenience, when i=0, the synaptic weight w_{j0} is associated with a fixed input x_0=-1 and is equal to the threshold (or bias) of the neuron j. Now, the partial derivative $\partial y_j(n)/\partial w_{ji}(n)$ in Equation (4) can be obtained as

$$\frac{\partial y_j(n)}{\partial w_{ji}(n)} = \frac{\partial f[w_{ji}(n), x_i(n)]}{\partial w_{ji}(n)}. \qquad (7)$$

If the transfer function $f[w_{ji}(n), x_i(n)]$ of the nonlinear synapse is known, this partial derivative can be easily evaluated. Making use of Eq. (4), (5), and (7) in Eq. (3), the increment $\Delta w_{ji}(n)$ can be expressed as

$$\Delta w_{ji}(n) = \eta e_j(n) \frac{\partial f[w_{ji}(n), x_i(n)]}{\partial w_{ji}(n)}, \qquad (8)$$

and the updated weight $w_{ji}(n+1)$ becomes

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \qquad (9)$$

Consider now that the neuron j is located in a hidden layer. In this case, there is no specified desired response for this neuron and the local error gradient would have to be determined recursively in terms of the local error gradients of all next layer neurons to which the hidden neuron is directly connected. Returning to Eq. (4), we note that the partial derivative $\partial y_j(n)/\partial w_{ji}(n)$ is still given by Eq. (7), but the local error gradient $\delta_j(n)$ has to be reevaluated. It can be easily shown that $\delta_j(n)$ of a hidden neuron j is expressed as

$$\delta_j(n) = -\sum_k \delta_k(n) \frac{\partial f[w_{kj}(n), y_j(n)]}{\partial y_j(n)}, \qquad (10)$$

where k denotes the index of neuron in the next layer fed by the hidden neuron j and $w_{kj}(n)$ is the associated weight. Thus, starting from the output layer, evaluating the local error gradients there (Eq. (5)), and propagating the errors backward, $\delta_j(n)$'s of all hidden layer neurons can be obtained recursively using Eq. (10). Finally, the weight increment $\Delta w_{ji}(n)$ is obtained as

$$\Delta w_{ji}(n) = -\eta \delta_j(n) \frac{\partial f[w_{ji}(n), x_i(n)]}{\partial w_{ji}(n)}. \qquad (11)$$

It is worth noting that the weights are updated either after every pattern presentation (on-line learning) or at the end of the epoch after all pattern presentations (off-line learning) by accumulating the weight increments. The learning cycle continues, until the total sum of squared errors becomes less than a specified value.

3. Modified node-perturbation learning algorithm with nonlinear synapses

In on-chip perturbation-based learning algorithms, the weight increment $\Delta w_{ji}(n)$ is still given by Eq. (3), but the error gradient $\partial E(n)/\partial w_{ji}(n)$ can be evaluated using alternative methods [6,7]. The simplest way is to perturb serially all the weights throughout the network using a small perturbation Δw_{pert} and measure the error $\Delta E(n)$ resulting from each perturbation. Thus $\Delta w_{ji}(n)$ can be approximately expressed as

$$\Delta w_{ji}(n) \cong -\eta \left.\frac{\Delta E(n)}{\Delta w_{pert}}\right|_{w_{ji}(n)} \qquad (12)$$

As mentioned previously, the SWP algorithm has an advantage that it makes no assumptions about the synapse and neuron characteristics and therefore, is extremely tolerant of nonidealities of analog circuits. However, the learning process is ver slow especially in the networks having large numbers of synapses. To speedup the learning process, an on-chip CHRP learning algorithm has been proposed [8]. In this algorithm, the error gradient $\partial E(n)/\partial w_{ji}(n)$ is given by Eq. (4) and can be approximated as

$$\frac{\partial E(n)}{\partial w_{ji}(n)} \cong \left.\frac{\Delta E(n)}{\Delta y_{pert}}\right|_{y_j(n)} \left.\frac{\Delta y_j(n)}{\Delta w_{pert}}\right|_{w_{ji}(n)}, \qquad (13)$$

where Δy_{pert} and Δw_{pert} are the perturbations applied to the neuron output $y_j(n)$ and the synaptic weight $w_{ji}(n)$, respectively. The outputs of the neurons in a given layer are serially perturbed and the corresponding values of $\Delta E(n)$ are measured. Then all weights originating from a given neuron i, in the previous layer, are perturbed in parallel and the corresponding values of $\Delta y_j(n)$ are measured. The simultaneous perturbations of a group of weights will result in a significant speedup of the learning process.

Now, if the transfer functions $f(w_{ji}, x_i)$ of nonlinear synapses are known, the above CHRP algorithm can be modified and a pseudo chip-in-loop supervised learning can be carried out. In fact, the local error gradient of the output layer neurons, can be obtained in a similar way as that used in the modified backpropagation algorithm and the output layer weights are updated directly with increments given by Eq. (8). On the other hand, for a hidden neuron, the quantity $\partial E(n)/\partial w_{ji}(n)$ can be approximately expressed as

$$\frac{\partial E(n)}{\partial w_{ji}(n)} \cong \left.\frac{\Delta E(n)}{\Delta y_{pert}}\right|_{y_j(n)} \frac{\partial f[w_{ji}(n), x_i(n)]}{\partial w_{ji}(n)} \qquad (14)$$

To be specific, the outputs of the neurons in a given hidden layer are sequentially perturbed and the corresponding changes in the error $\Delta E(n)$ are calculated. For each hidden neuron, the local error gradient $\Delta E(n)/\Delta y_{pert}$ is evaluated and then used in Eq. (14) to obtain $\partial E(n)/\partial w_{ji}(n)$ for all synaptic connections linking this neuron with all neurons in the previous layer. In this manner, the hidden neuron outputs have only to be perturbed and there is no need to perturb the groups of synaptic weights as in the case of CHRP algorithm. We call therefore this modified CHRP version, the modified Node Perturbation (NP) learning algorithm. Finally, having determined $\partial E(n)/\partial w_{ji}(n)$, the weight increment $\Delta w_{ji}(n)$ can again be computed using Eq. (3) and an iterative process (either on-line or off-line is carried out to obtain the final updated weights satisfying a specific error criterion.

4. Compact synapse-neuron circuit

Conventional MLFFNNs are composed of layers of neurons that apply nonlinearity to the sum of previous layer weighted outputs. The key elements of hardware implementation of neural networks are the synapse and neuron circuits. An analog synapse generally generates an output current that is proportional to the product of its input and weight. A neuron converts the sum of the synapse currents to a voltage and applies nonlinearity. This will always result in a synapse-neuron unit of a relatively large size, since a wide-range multiplier followed by a

current-to-voltage converter and a sigmoid generator have to be employed. This topology is inspired by the biological neuron model as a crude attempt to model complex biological neural systems. However, since the important characteristic of a neuron model is its ability to perform nonlinear mappings [10], nonlinear synapses may posses this characteristic. Fig.1 shows a compact synapse-neuron unit employed in the present work. Instead of using a wide-range linear multiplier, which is in general, composed of a Gilbert-multiplier cell, an attenuator and a level-shifter [4], a more compact Gilbert-cell can only be used. An example of a MOS version of such a type of four-quadrant multipliers is shown in Fig. 1-a. This cell consists of three source-coupled pairs, two of which are cross-linked and composed of the matched transistors M1, M2 and M3, M4, respectively. The third source-coupled pair is composed of the matched transistors M5 and M6. M7 is connected as a constant current sink to provide the required tail current. The input signals of the cell are V_x and V_w and represent the neuron input and the associated synaptic weight, respectively. Considering strong inversion operation and assuming that all transistors operate in saturation, the differential output current I_{od} is given by

$$I_{od} = I_{o1} - I_{o2}$$

$$= \sqrt{\beta_a} V_x \left[\sqrt{I_6 - \frac{\beta_a V_x^2}{4}} - \sqrt{I_5 - \frac{\beta_a V_x^2}{4}} \right], \qquad (15)$$

where $\beta_a = \mu_o C_{ox} (W/L)_a$, is the transconductance parameters of transistors M1-M4 (μ_o being the channel mobility, C_{ox} being the gate oxide capacitance per unit area and (W/L) being the transistor aspect ratio). The currents I_5 and I_6 are the drain currents of M5 and M6, respectively and are related to the differential voltage V_w by

$$\sqrt{I_6} - \sqrt{I_5} = \sqrt{\frac{\beta_b}{2}} V_w \ , \qquad (16)$$

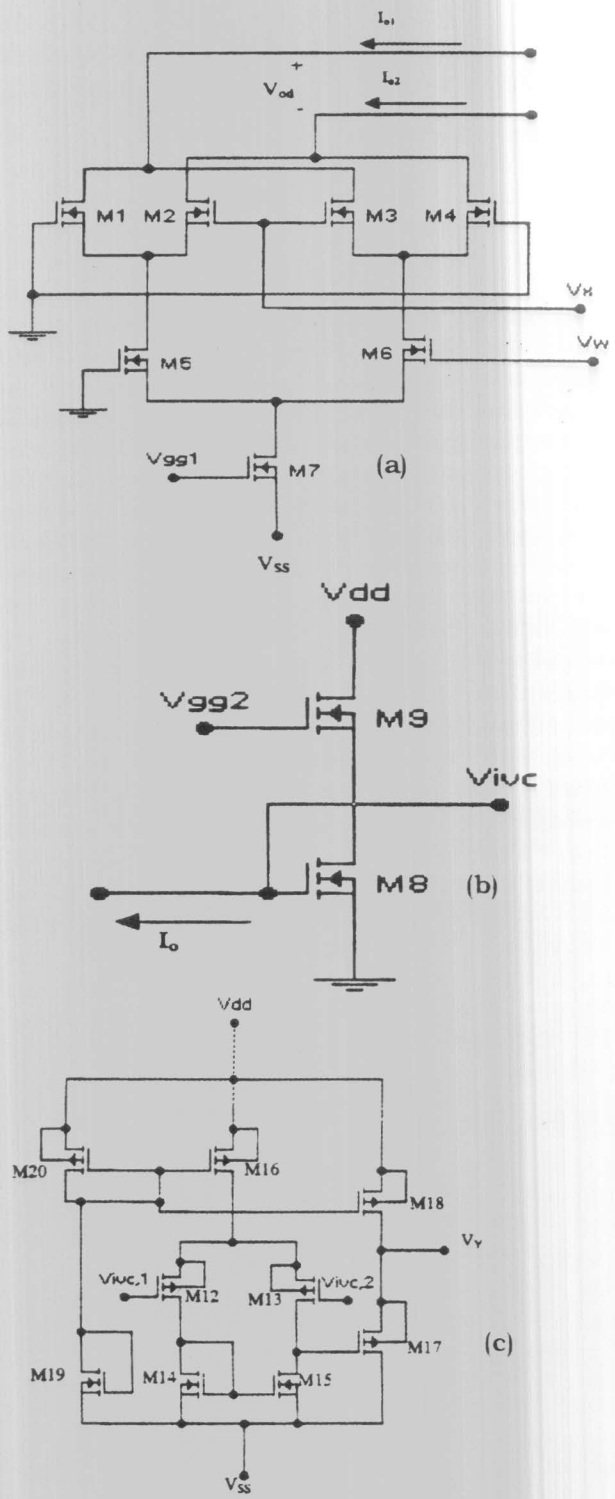

Fig. 1. Compact synapse-neuron unit. (a) Four-quadrant Gilbert multiplier. (b) NMOS-IVC (two converters of such a type are used). (c) Differential-input single-output CMOS amplifier stage.

where $\beta_b = \mu_o C_{ox} (W/L)_b$, is the transconductance parameter of transistors M5 and M6. Now, if $V_x \ll \sqrt{4I_{5,6}/\beta_a}$, then I_{od} can be approximated as

$$I_{od} = \sqrt{\frac{\beta_a \beta_b}{2}} V_x V_w \quad (17)$$

It is worth noting that as long as the above condition validating Eq.(17) is satisfied and that all transistors remain operating in saturation, a linear multiplication process is performed, otherwise a nonlinear behavior takes place. Fig. 2 illustrates a SPICE simulation for the Gilbert-cell of Fig. 1-a. The transistor aspect ratios and the MOSFET SPICE parameters (2 μm-CMOS technology) are similar to those used in [4]. For the simulation purpose, the output nodes are connected to load resistors R_L= 20 KΩ and the differential output voltage $V_{od} = I_{od} R_L$ is plotted as a function V_x for different values of V_w in the range between −2V and 2V. It is noted that the cell is performing linear multiplication for differential inputs smaller than about ±0.2V. For larger differential inputs, the output tends to saturate and the cell exhibits nonlinearity. In the practical realization of a complete MLFFNN, the differential output currents of different synapse units can be summed by hardwiring the corresponding output nodes of these synapses. The current sum is converted to a voltage using a current-to-voltage converter (IVC) and then amplified, resulting in the neuron output voltage V_y. An example of NMOS-IVC is shown in Fig. 1-b [4]. In this configuration, both M8 and M9 are ON and operate in saturation provided that $2V_{Tn} < V_{GG2} \le V_{DD}$; V_{Tn} being the threshold voltage of the NMOS devices. It can be shown that the converter output voltage is expressed as

$$V_{IVC} = \frac{I_o}{\beta(V_{GG2} - 2V_{Tn})} + \frac{V_{GG2}}{2}, \quad (18)$$

where $\beta_8 = \beta_9 = \beta$. Since the synapse output current is in a differential form, two IVCs have to be used. The term $V_{GG2}/2$ in Eq. (18)

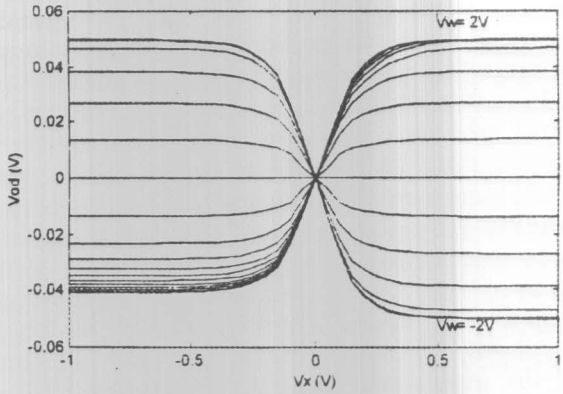

Fig. 2. SPICE simulation of the Gilbert cell (see text).

presents an offset and can be cancelled out using a high-gain differential amplifier. The outputs of the two IVCs are applied to differential amplifier inputs and the amplifier output will be then proportional to the sum of synapse differential output currents. Fig. 1-c shows a CMOS differential-input single-ended output amplifier, employed in the present work [4]. The amplifier consists of a differential stage composed of transistors M12-M16, followed by an output stage (source follower, M17 and M18). M19 and M20 are used for bias purpose. A SPICE simulation of the IVCs and the differential amplifier is shown in Fig. 3. Good linearity has been achieved for the specified range of synapse differential output current. Thus, the overall nonlinearity desired from the compact synapse-neuron unit will essentially come from the synapse cell. A SPICE simulation of this compact unit is shown in Fig. 4, where the input V_x s aried etween 1V nd V nd the weight V_w is kept constant at fixed values ranging from −2V and 2V. It is worthwhile noting that the compact unit contains only 20 MOS devices while the conventional synapse-neuron unit, employing a wide-range multiplier as a synapse and an IVC and a sigmoid generator as a neuron, contains about 40 MOS devices [4]. This considerable reduction in the number of MOS devices will lead to corresponding reductions in the chip-area and the power consumption of the realized complete neural network, provided that learning algorithms tailored for such a type of networks are employed. The modified backpropagation and node-perturbation

learning algorithms discussed in Sections (2) and (3) are good candidates for these networks. The synapse-neuron transfer functions obtained in Fig. 4 can be either fitted by mathematical functions (e.g, a polynomial function of two variables V_x and V_w) or tabulated in a look-up table after being quantized, to obtain $V_y = f(V_x, V_w)$. These fitting or tabulated transfer functions can be then implemented in software programs based on the modified learning algorithms for pseudo chip-in-loop training. In the present work, the transfer functions are modeled by a look-up table of size 81×81 where the inputs and weights (varying between -2V and 2V) are quantized to 81 different values, centered at 0V (i.e, with a quantization step of 0.05V). The closest value is chosen when the inputs or weights do not match the values in the table. Derivatives are evaluated from finite differences. After training, the final updated weights are downloaded to the hardware version of the network for feedforwared SPICE simulations.

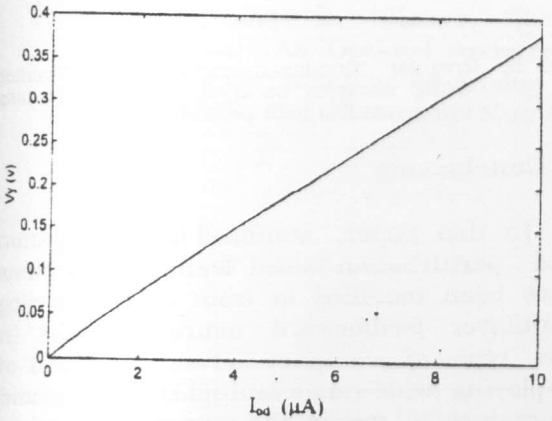

Fig. 3. Transfer characteristic (SPICE Simulation) of IVCs and differential amplifier.

5. Algorithms and training approach validation

The validity of the proposed learning algorithms and the pseudo chip-in-loop training approach have been tested by designing complete MLFFNNs with different architectures and training them to perform different tasks.

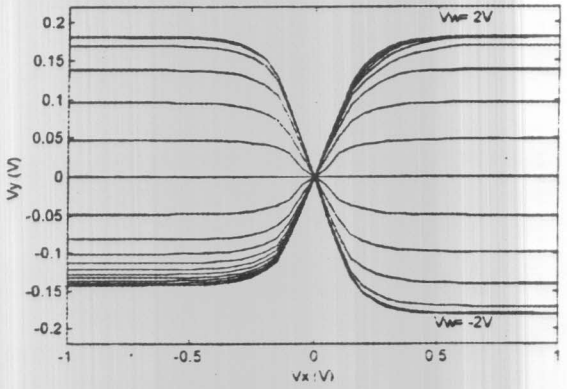

Fig. 4. Overall transfer characteristics (SPICE Simulation) of the compact synapse-neuron unit.

An example of such tasks performed successfully by MLFFNNs, is function approximation. In this application, it is noted that a network with one input node, one hidden layer containing a relatively small number L of neurons, and one output neuron (i.e, 1:L:1 architecture), can be used [11]. In the present work, a 1:5:1 network has been used. The network (Fig. 5) has a hidden layer of 5 neurons, each of which is connected to the input node through a synaptic connection. This layer can be constructed from 5 nonlinear multipliers (Gilbert-cells), 5 IVC, and 5 differential amplifiers. The output layer has a single neuron with 5 synaptic connections from the hidden neurons and can be constructed from 5 Gilbert-cells, one IVC and one differential amplifier. It is worth at this point to compare the number of MOS devices used in this compact network with that used in networks employing conventional neuron models (i.e, linear multipliers and sigmoid generators). In the present compact version, 148 MOS devices are used in comparison with 288 devices in the conventional network version of Ref. [4]. Thus, roughly, a reduction of about 50% in the chip-area and consequently in the power consumption, may be expected.

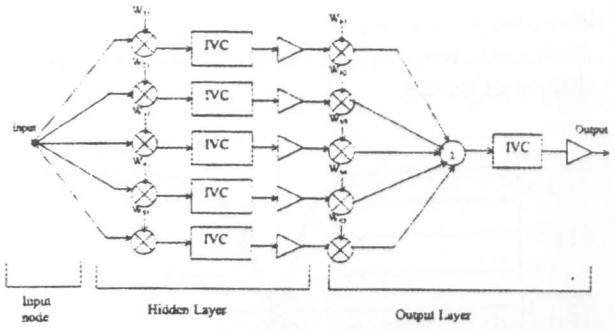

Fig. 5. Block diagram of a compact 1:5:1 network used in function approximation.

An example of a function approximation is illustrated in Fig. 6, where the network is trained to approximate an irregular function [11]. The training set contains 17 input/target pairs (symbols "*") in the interval [-2V , 2V]. The training procedure is performed using both the modified backpropagation and node-perturbation learning algorithms and the final updated weights are downloaded to both the software and hardware versions of the network for feedforward simulations. It is noted that some network parameters including learning rate η, perturbation magnitude Δy_{pert} and gain of synapse-neuron unit, are not optimized for minimum number of epochs satisfying a specific error criterion. This point will be studied and published elsewhere. In the present work, comparable numbers of epochs have been obtained for the two learning algorithms. The solid curves in Figs. 6-a and 6-b represent the responses of the software networks after being trained using the modified backpropagation and node-perturbation learning algorithms, respectively. The "o" symbols in these figures denote the responses of the trained hardware networks (SPICE simulations) in both cases. Good agreement between the software and hardware network responses and the desired response has been achieved. This in fact, reflects the efficiency and accuracy of the modified learning algorithms and the pseudo chip-in-loop training approach employed in this work, especially when accurate representation (high density look-up table) of synapse-neuron transfer functions, is taken into account.

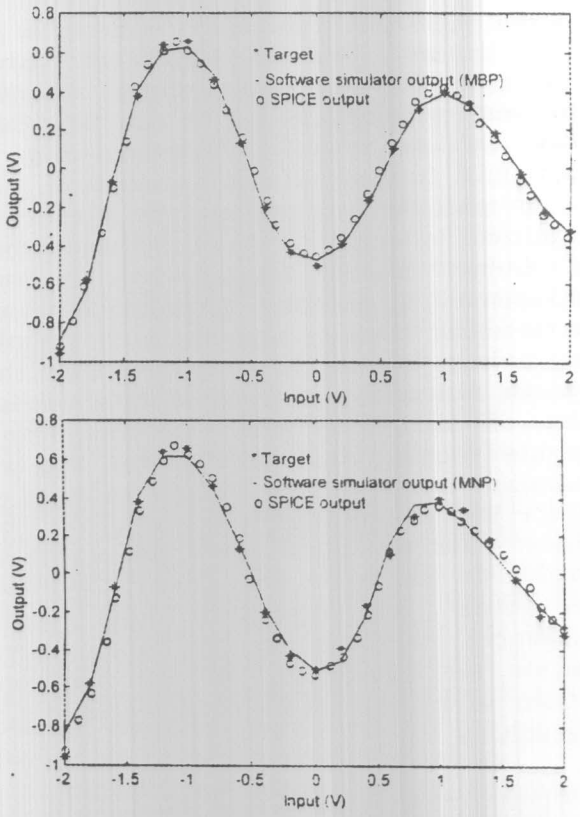

Fig. 6. Irregular function approximation. (a) Training using modified backpropagation. (b) Training using modified node perturbation.

6. Conclusions

In this paper, standard backpropagation and perturbation-based learning algorithms have been modified to train compact analog multilayer feedforward neural networks. In this type of compact networks, instead of employing wide-range multipliers and sigmoid generators to mimic the synapse and neuron functions, a much more compact synapse-neuron unit, a nonlinear multiplier, has been used. The compact unit has been simulated and the resulting transfer functions have been implemented in software programs based on the modified learning algorithms. After training, the final updated weights are downloaded to the network for feedforwared operation. To test the validity of modified learning algorithms and training approach a complete network of such a compact type has been designed and trained to perform successfully a function approximation task.

References

[1] S. Haykin, "Neural Networks: a Comprehensive Foundation", 2nd Edition, Macmillan (1999).

[2] T. Shima, T. Kimura, Y. Kamatani, T. Itakura, Y. Fujita, and T. Iida, "Neuro Chips with On-Chip Back-Propagation and/or Hebbian Learning", IEEE J. Solid-State circuits, Vol. 27, pp. 1868-1876 (1992).

[3] T. Morie and Y. Amemiya, "An All-Analog Expandable Neural Network LSI with On-Chip Backpropagation learning", IEEE J. Solid-State Circuits, Vol. 29, pp. 1086-1093 (1994).

[4] Y. Akl and M. Elsayed . "Pseudo Chip In-Loop Supervised Learning for Analog VLSI Neural Networks", Alexandria Engineering Journal, Vol. 38, pp. 65-78 (2000).

[5] M. Jabri, R. Coggins, and B. Flower, "Adaptive Analog VLSI Neural Systems", Chapman and Hall (1996).

[6] M. Jabri and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks", IEEE Trans. Neural Networks, Vol. 3, pp. 154-157 (1992).

[7] P. Hollis and J. Paulos, "A Neural Network Learning Algorithm Tailored for VLSI Implementation", IEEE Trans. Neural Networks, Vol. 5, pp. 784-791 (1994).

[8] A. Montalvo, R. Gyurcsik, and J. Paulos, "An Analog VLSI Neural Network with On-Chip Perturbation Learning", IEEE J. Solid-State Circuits, Vol. 32, pp. 535-543 (1997).

[9] A. Cichocki and R. Unbehauen, "Neural Network for Optimization and Signal Processing", John Wiley (1995).

[10] P. Hollis and J. Paulos, "Artificial Neural Networks using Analog Multipliers", IEEE J. Solid-State Circuits, Vol. 25, pp. 849-855 (1990).

[11] The math works Inc. "Neural Network Toolbox for use with MATLAB"

Received February 29, 2000
Accepted May 3, 2000