14 ANALOG VLSI ON-CHIP LEARNING NEURAL NETWORK WITH LEARNING RATE ADAPTATION

Gian Marco Bo, Daniele D. Caviglia, Hussein Chiblé, and Maurizio Valle

Department of Biophysical and Electronic Engineering
University of Genoa
Via all'Opera Pia 11/A, 16145 Genova, Italy
{gian.bo, caviglia, valle}@dibe.unige.it

This chapter describes an analog VLSI implementation of a multilayer perceptron neural network with on-chip backpropagation learning. Local adaptation of the learning rate offers fast convergence. Experimental results from a chip fabricated in the ATMEL ES2 ECPD07 0.7μm CMOS process demonstrate learning of a 2-input exclusive-or task in 32 ms.

14.1 INTRODUCTION

In this chapter the analog VLSI implementation of a Multi Layer Perceptron (MLP) network with on-chip learning capability is presented. A MLP architecture is chosen since it can be applied to successfully solve real-world tasks, see among others [33, 6, 9, 4].

Many examples of analog implementations of neural networks with on-chip learning capability have been presented in literature, for example [34] [3][35] [40] [41] [14] [28] [38] [5] [24]. Learning speeds in many of these implementations are rather slow, ranging from milliseconds with volatile weight storage to seconds with non-volatile storage. Improvements in learning speed require not only dedicated circuit design, but also a careful design of the algorithms and architecture used. This chapter demonstrates the advantages of a local learning rate adaptation algorithm and corresponding hardware implementation for efficient on-chip learning.

Design issues regarding analog implementation of on-chip Back Propagation (BP) have been discussed in detail, e.g., in [2, 17, 7]. The learning rate [19] is a critical parameter in learning performance, which has motivated us to consider techniques for local adaptation of learning parameters, the learning rate in particular [8], thus
improving the speed of training convergence. It has also been demonstrated that adaptation of the learning parameters is beneficial to the fault tolerance of the learned neural network [18].

The next section starts with a brief review of the implemented multilayer perceptron (MLP) model, and backpropagation learning. We then present the algorithmic modification with local learning rate adaptation in Section 14.3, and the analog VLSI implementation in Section 14.4. Sections 14.5 and 14.6 conclude with experimental results from a fabricated MLP prototype including the on-chip learning rate adaptation.

14.2 NEURAL PARADIGM AND LEARNING ALGORITHM

14.2.1 The feed-forward network

Figure 14.1 shows the standard architecture of a feed-forward $N$-layer MLP network [32, 19].

![Diagram of a multi-layer perceptron network](image)

Figure 14.1 Standard architecture of a $N$-layer Multi Layer Perceptron (MLP) network.

The $l^{th}$ layer ($l = 1, \ldots, N$) consists of a matrix of $n^l \times n^{l-1}$ synaptic multipliers (indicated by arrows) and a vector of $n^l$ neurons (indicated by "+" blocks). The neurons implement the sum of the synaptic outputs and a non-linear activation function $\Psi()$. The neurons usually have output values in the range $[-1, +1]$, and the non-linear activation function $\Psi()$ is the hyperbolic tangent. The computation performed by the $j$th neuron of the $l^{th}$ layer, for $j = 1, \ldots, n^l$ and $i = 1, \ldots, n^{l-1}$, is given by:

$$a^l_j = \sum_{i=1}^{n^{l-1}} W^l_{j,i} \cdot X^{l-1}_i$$

$$X^l_j = \tanh(a^l_j)$$

(14.1)

where $X^l_j$ is the output of the $j^{th}$ neuron of the $l^{th}$ layer, $X^{l-1}_i$ is the output of the $i^{th}$ neuron of the $(l-1)^{th}$ layer, the term $a^l_j$ (usually called activation) is the sum of the outputs of the synapses of the $(l-1)^{th}$ layer connecting to the $j^{th}$ neuron of the $l^{th}$ layer, and $W^l_{j,i}$ is the weight of the synapse connecting the $j^{th}$ neuron of the $l^{th}$ layer to the $i^{th}$ neuron of the $(l-1)^{th}$ layer. The input layer, neuron layer 0, does not
perform computation but feeds the input signals, \( i = 1, \ldots, n^0 \) to the network. The bias terms, not explicitly shown in Figure 14.1 and Equation (14.1), are usually considered as weight connections between each neuron and an input whose value is held fixed at 1, so the formulation given is general.

14.2.2 The Learning Algorithm

We define the error function \( E_p \) for the \( p \)th pattern example as the squared difference between the target network output \( X_k^N \) and the actual network output \( X_k^N \):

\[
E_p = \frac{1}{2} \sum_{k=1}^{n^N} (X_k^N - X_k^N)^2
\]

where the index \( k \) runs over the output layer neurons.

The error function \( E \) is computed as the average of the error functions \( E_p \) over all the pattern examples of the training set

\[
E = \frac{1}{N_p} \sum_{p=1}^{N_p} E_p
\]

where \( N_p \) is the number of pattern examples of the training set.

The training of the MLP network is achieved by minimizing the total error function \( E \) with respect to the weight values. This task is accomplished using gradient-descent with the Back Propagation (BP) algorithm [32, 19]. We consider the BP rule and a by-pattern example presentation approach. Starting from an initial random weight guess configuration, at each pattern presentation ("learning iteration" in the remainder of the chapter) the weights values are updated according to a rule following which the trajectory in the weight space is along the negative gradient direction of the error function \( E_p \) [32, 19, 15]:

\[
\Delta W_{j,i} = -\eta \frac{\partial E_p}{\partial W_{j,i}}
\]

where \( \eta \) is the learning rate, a positive parameter. The value of \( \eta \) strongly affects the learning procedure: if \( \eta \) is sufficiently small, iteration of the procedure converges to a (local) minimum of the total error function \( E \). In comparison to the by-epoch approach, the by-pattern procedure introduces randomness in the learning process that often helps in escaping local minima of the total error function \( E \). Moreover, this technique is usually faster and more effective when the training set is very large, composed of thousands of pattern examples as in the case of hand-written character recognition, speech recognition, etc. By applying the chain rule to Equation (14.4), we can obtain the following learning rule:

\[
\Delta W_{j,i} = \eta \cdot \delta_j \cdot X_i^{i-1}
\]

\[
\delta_j^N = -\frac{\partial E_p}{\partial X_j^N} = (X_k^N - X_j^N) \cdot D_j^N
\]

\[
\delta_j = -\frac{\partial E_p}{\partial \delta_j} = \left( \sum_{k=1}^{n^{i+1}} \delta_k \cdot W_{k,j}^{i+1} \right) \cdot D_j^i
\]
where:

\[ D_j^t = 1 - (X_j^t)^2 \]  \hspace{1cm} (14.6)

### 14.2.3 Improvements in the Learning Convergence Speed

Some approaches to increase the convergence speed are possible:

1. estimation of the optimal learning rate value [30];

2. reduction of the input data space dimension by pre-processing the data (e.g., [12] for character recognition);

3. use of sophisticated and complex gradient descent algorithms, e.g., usage of the momentum term [15]).

Among other approaches, one can consider, instead of a single learning rate \( \eta \), an individual learning rate for each synapse, \( \eta_{j,i} \). The local learning rate value can be locally adapted following some heuristics, and the weight update rule becomes:

\[
\Delta W_{j,i}^t = \eta_{j,i}^t \cdot \delta_j^t \cdot X_j^{t-1} \]  \hspace{1cm} (14.7)

Some approaches have been presented in literature [22, 36] to adapt the local learning rate values. The rationale can be summarized as follows [15]:

1. If the component \( \partial E_p / \partial W_{j,i}^t \) of the gradient of the error function \( E_p \) retains the same sign for many consecutive iterations, the corresponding learning rate \( \eta_{j,i}^t \) is increased since a local minimum lies in such direction;

2. If the sign of \( \partial E_p / \partial W_{j,i}^t \) changes for consecutive iterations, the corresponding learning rate is decreased, since this indicates that a minimum is nearby, and missed (jumped over) by too coarse learning steps.

Since the learning rate value is critical for achieving the convergence and for determining the learning speed, i.e., number of iterations needed to train the network, local adaptation of the learning rates improves the network performance [22, 36, 18]. In [37] we introduced a learning algorithm, inspired by [22] and [36], based on local synaptic learning rate adaptation according to the signs of gradient components. The algorithm was validated in training analog MLPs in a "chip-in-the-loop" configuration [2]. The local learning rate adaptation rule we adopt here is derived from [37] and it is described by Equation (14.8):

\[
\begin{align*}
\text{if} \quad S_{j,i}^t &= S_{j,i}^{t-1} \\
\eta_{j,i}^t(t + 1) &= \eta_{j,i}^t(t) \left( \frac{\eta_{\text{max}}}{\eta_{j,i}^t(t)} \right)^\gamma \\
\text{else} \\
\eta_{j,i}^t(t + 1) &= \eta_{j,i}^t(t) \left( \frac{\eta_{\text{min}}}{\eta_{j,i}^t(t)} \right)^\gamma
\end{align*} \hspace{1cm} (14.8)
\]
where, taking into account the generic synapse connecting the $j^{th}$ neuron of the $l^{th}$ layer and the $i^{th}$ neuron of the $(l - 1)^{th}$ layer, $S_{j,i}^l(t)$ is the sign of the gradient component $\partial E_p/\partial W_{j,i}^l$, and $\eta_{j,i}^l$ is the learning rate value at the $t^{th}$ iteration; $\eta_{max}$ and $\eta_{min}$ are respectively the maximum and minimum values of the learning rate and $\gamma$ is the learning rate adaptation coefficient $\gamma \in [0, 1]$. Some further considerations for the adaptation rule (14.8) are:

1. each synapse has its learning rate locally adapted according to the sign of the gradient component;
2. the computation of Equation (14.8) is easily mapped onto analog circuits, as explained in Section 14.4;
3. the adaptation rule can be extended to many other learning algorithms to train feedforward and recurrent neural networks (see, among others, [1, 10], where model-free algorithms are used to perform the training operations).

14.3 ANALOG ON-CHIP LEARNING ARCHITECTURE

The analog architecture of a MLP neural network with on-chip BP learning is shown in Figure 14.2 [38].

![Figure 14.2](image)

Figure 14.2 The analog on-chip BP learning MLP architecture.

We consider two neuron layers, $N = 2$, since it has been demonstrated that this network is able to perform arbitrary non-linear mappings between two data spaces if the network has a sufficient number of neurons in the first layer, $l = 1$ [32, 19]. The proposed architecture can be easily extended to the case $N > 2$. The hardware
architecture consists of two matrices of synaptic modules indicated by $S$, and two arrays of neuron modules indicated by $N$. We refer to the synapses in the first and second layer as hidden and output synapses respectively. In the same way we refer to neurons in the first and second layer as hidden and output neurons. The layer-index is not shown in Figure 14.2 for the sake of simplicity; index $k$ refers to the output layer ($l = 2$), index $j$ to the hidden layer ($l = 1$), and index $i$ refers to the input layer ($l = 0$).

The output synaptic modules have four terminals: the synaptic input $X_j$, the synaptic output $W_{k,j} \cdot X_j$, the input error term $\delta_k$ and the backpropagating error term $\delta_k \cdot W_{k,j}$.

The output neuron modules have also four terminals: the neuron input $\Sigma_h W_{k,j} \cdot X_j$, the neuron output $X_k$, the target $X_i$, and the error terms $\delta_k$. The hidden synaptic modules contain similar modules: synaptic input $X_i$, the synaptic output $W_{j,i} \cdot X_i$, the input error term $\delta_j$, and no backpropagating error signal since there is no preceding module.

We can identify in Figure 14.2 two different orthogonal signal paths related to the feed-forward phase, Equation (14.1), and learning back-propagation phase, Equation (14.6). The structure of the synaptic and neuron modules are discussed in the next subsections.

### 14.3.1 The synaptic module

![Block diagram of the synaptic module](image)

The block diagram of the synaptic module is shown in Figure 14.3. It contains the following blocks:

1. $F$ is the feed-forward four-quadrant multiplier, performing multiplication between the synaptic input $X_j$ and the weight value $W_{k,j}$.

2. $B1$ is the backward four-quadrant multiplier, performing multiplication between the error term $\delta_k$ and the weight $W_{k,j}$.

3. $B2$ is the weight update four-quadrant multiplier, performing the multiplication between the synaptic input $X_j$, the error term $\delta_k$, and the learning rate $\eta_{k,j}$, thus generating the weight update signal $W_{k,j} = \eta_{k,j} \cdot \delta_k \cdot X_j$. Block $B2$ also generates the sign $S_{k,j}$ of the corresponding component of the gradient of $E_p$: $S_{k,j} = \text{sign}(\partial E_p/\partial W_{k,j}) = -\text{sign}(\Delta W_{k,j})$.
4. $H$ is the local learning rate adaptation circuit block, adapting the value of $\eta_{k,j}$ according to the sign of two consecutive weight update values, Equation (14.8);

5. $WU$ is the weight block, updating the weight values $W_{k,j}$ according to $W_{k,j}^{\text{new}} = W_{k,j}^{\text{old}} + \Delta W_{k,j}$. The $WU$ block performs also short-term memory storage of the weight value.

As stated before, in the hidden synaptic modules the multiplier $B1$ is not required since the modules do not backpropagate any error signal.

### 14.3.2 The neuron module

![Block diagrams of the output (a) and hidden (b) neuron modules.](image)

The structure of the neuron module is shown in Figure 14.4. The structure of the output neuron module in Figure 14.4 (a) is slightly different with respect to the one of the hidden neuron module of Figure 14.4 (b). The neuron module contains the following blocks:

1. $A$, activation function module, implementing the activation function $\Psi(a)$, i.e., $X_{k(j)} = \Psi (a_{k(j)})$;

2. $D$, derivative module, computing the derivative of the activation function i.e., $D_{k(j)} = 1 - (X_{k(j)})^2$;

3. $R$, the error multiplier, computing the error term by multiplying the derivative of the activation function by the term $(X_{k} - X_{k})$ or $\delta_{k} = (X_{k} - X_{k}) \cdot D_{k}$ in the output neuron and $\delta_{j} = (\Sigma_{k} \delta_{k} \cdot W_{k,j}) \cdot D_{j}$ in the hidden neuron (see Figure 14.4);

4. $FC$, the error circuit, computing the quadratic error between the target $X_{k}$ and the network output $X_{k}$, $\epsilon_{k} = (X_{k} - X_{k})^2$. The error function can be obtained by summing the quadratic error related all the output neurons, $\Sigma_{k} \epsilon_{k}$. 
Table 14.1 Correspondence between the neural and electrical variables for the synaptic and neuron modules

<table>
<thead>
<tr>
<th>Synaptic module</th>
<th>Neuron module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural variables</td>
<td>Electrical variables</td>
</tr>
<tr>
<td>$X_i$</td>
<td>$V_X$</td>
</tr>
<tr>
<td>$W_{k,i} \cdot X_j$</td>
<td>$I_{WX}$</td>
</tr>
<tr>
<td>$W_{k,j}$</td>
<td>$V_W$</td>
</tr>
<tr>
<td>$\Delta W_{k,j}$</td>
<td>$I_{\Delta W}$</td>
</tr>
<tr>
<td>$\delta_k$</td>
<td>$V_\delta$</td>
</tr>
<tr>
<td>$\eta_{k,j}$</td>
<td>$I_\eta$</td>
</tr>
<tr>
<td>$S_{k,j}$</td>
<td>$V_{SN}$</td>
</tr>
<tr>
<td>Neuronal variables</td>
<td>Electrical variables</td>
</tr>
<tr>
<td>$a_{k(i)}$</td>
<td>$I_a$</td>
</tr>
<tr>
<td>$X_{k(i)}$</td>
<td>$V_X$</td>
</tr>
<tr>
<td>$X_{k(i)}$</td>
<td>$V_T$</td>
</tr>
<tr>
<td>$\sum_k \delta_k \cdot W_{k,i}$</td>
<td>$V_{bw}$</td>
</tr>
<tr>
<td>$\delta_{k(j)}$</td>
<td>$V_\delta$</td>
</tr>
<tr>
<td>$D_{k(j)}$</td>
<td>$I_d$</td>
</tr>
<tr>
<td>$\epsilon_k$</td>
<td>$I_e$</td>
</tr>
</tbody>
</table>

14.3.3 Mapping neural variables onto analog signals

The variables of the algorithm need to be coded into electrical signals such as voltages, currents, or charges. Normally, current signals are used for summing variables, exploiting the Kirchoff Current Law (KCL), while voltage signals are used for distribution across modules. Table 14.1 lists the coding used for neural signals in the circuit implementation of the synaptic module (a) and neuron module (b). The supply voltage $V_{dd}$ was set to 5V, and the signal ground was set to $V_{dd}/2$, 2.5V. The dynamic range of the neural signal voltages is limited to [2.4V, 2.6V], for reasons explained in the sections below. No a-priori limits are present for the current values; the only constraint is that current values must correctly bias the MOS transistors.

14.4 CMOS ON-CHIP LEARNING IMPLEMENTATION

In Section 14.3 the analog architecture for the hardware implementation of a MLP network with on-chip BP learning and local learning rate adaptation, based on the algorithm introduced in Section 14.2, was introduced. The neural paradigm and the on-chip learning algorithm have been mapped onto the analog architecture by considering as main computational primitives the synaptic and neuron modules. Their block diagrams are described in detail in Section 14.3. The circuit implementation is based partially on the one proposed in [38]. To reduce the overall silicon area and power consumption we choose to design the circuits to operate, where possible, in the weak inversion region [27, 39]. Low values of bias currents for the neural circuits (e.g., less than 200 nA) allow to reduce the power consumption and the sizes of MOS transistors working in weak inversion. On the other hand, high values of bias currents (e.g., higher that 5 μA) improve the circuit transient performance, at the expense of an increase in power consumption, and larger-sized MOS transistors in order to stay close to the weak inversion region. Bias current values govern then the trade-off between response time and area/power consumption. We choose to design the neural computational circuits with current values in the range of 1μA.
14.4.1 The synaptic circuit

$F$ and $B1$ four-quadrant multiplier. The multiplier inputs are mapped onto voltages and the output is mapped onto a current; the multiplication is performed on four quadrants. No linear transfer characteristic with respect to $W_{k,j}$ is requested by such multiplier; the only constrain is that it must be a monotonic transfer characteristic [17, 38, 13]. The circuit implementing the feed-forward multiplier is shown in Figure 14.5. It is composed of two blocks: the $W$ block and the OTAs block.

![The feed-forward multiplier $F$ circuit.](image)

The $W$ block is a non-linear transconductor that converts the weight voltage $V_w$ into a differential current $I_w = I_{wp} - I_{wn}$. With equal aspect ratios $W/L$ for transistor pairs $M1$-$M2$ and $M3$-$M4$, and all transistors biased in strong inversion, we can write [13]:

$$I_{wn} = \begin{cases} \beta_n \cdot (V_w - V_{th1} - V_{th2})^2, & V_w \geq V_{th1} + V_{th2} \\ 0, & V_w < V_{th1} + V_{th2} \end{cases}$$

$$I_{wp} = \begin{cases} \beta_p \cdot (V_w - V_{th3} - V_{th4})^2, & V_w \leq V_{dd} + V_{th3} + V_{th4} \\ 0, & V_w > V_{dd} + V_{th3} + V_{th4} \end{cases}$$

where

$$\frac{1}{\sqrt{\beta_n}} = \frac{1}{\sqrt{\beta_1}} + \frac{1}{\sqrt{\beta_2}}, \quad \frac{1}{\sqrt{\beta_p}} = \frac{1}{\sqrt{\beta_3}} + \frac{1}{\sqrt{\beta_4}}$$

and $\beta_i$ and $V_{thi}$ are the gain factors and the threshold voltages of $M_i$ ($i = 1, \ldots, 4$) respectively [39]. The resulting differential current $I_w$ can be written as:

$$I_w = I_{wp} - I_{wn} = g_w(V_w)$$

(14.10)

where $g_w(V_w)$ is defined by (14.9). The OTAs block is composed of two Operational Transconductance Amplifiers (OTAs) biased in weak inversion. The synaptic output
current $I_{WX}$ can be expressed as [27, 39]:

$$I_{WX} = (I_{wp} - I_{wn}) \cdot \tanh \left( \frac{V_X - X_r}{2nU_T} \right) = g_w(V_W) \cdot \tanh \left( \frac{V_X - X_r}{2nU_T} \right)$$

(14.11)

where $n$ is the weak inversion slope coefficient [39], $U_T$ is the thermal voltage, and $V_r$ is the signal ground (the synaptic input is null for $V_X = V_r$). If the value of the argument of the tanh function is small, $|V_X - V_r| \leq 100 \text{ mV}$, we can approximate it with its argument [27]:

$$I_{WX} \approx \frac{1}{g_w(V_W)} \cdot (V_X - V_r)$$

(14.12)

The circuit of the $B1$ multiplier is equal to the one shown in Figure 14.5: one must substitute the synaptic input voltage $V_X$ with the error term voltage $V_{6i}$ and the synaptic output current $I_{WX}$ with $I_{6W}$ (see Figure 14.3 and Table 14.1).

The $B2$ four-quadrant multiplier. The input $x_k$ and $X_j$ of the multiplier are mapped onto voltages, while $\eta_{h,j}$ and the weight update $\Delta W_{h,j}$ are mapped onto currents. The sign $S_{h,j}$ can be computed from the sign of $\Delta W_{h,j}$: it is coded by a digital value, $S_{h,j} = \{0, V_{dd}\}$. The $B2$ multiplier is shown in Figure 14.6: it is basically a Gilbert multiplier [27] plus an additional output stage (transistors $M14$ and $M17$) that computes $S_{j,i}$.

![Figure 14.6 The B2 multiplier circuit.](image-url)
With transistors M1, M2, M7, M8, M9, and M10 biased in weak inversion, the current $I_{\Delta W}$ is expressed as [27]:

$$I_{\Delta W} = I_\eta \cdot \tanh \left( \frac{V_X - V_T}{2nU_T} \right) \cdot \tanh \left( \frac{V_\delta - V_T}{2nU_T} \right)$$  \hspace{1cm} (14.13)

and if $|V_X - V_T|$ and $|V_\delta - V_T| \leq 100mV$, we can approximate the tanh functions with their arguments [27]:

$$I_{\Delta W} \approx \frac{I_\eta}{4n^2U_T^2} \cdot (V_X - V_T) \cdot (V_\delta - V_T)$$  \hspace{1cm} (14.14)

Since transistors M14 and M17 drive a MOS gate with high output impedance, they act as a current comparator [31]. The voltage signal $V_S$ is high or low depending on the versus of the weight update current $I_{\Delta W}$, depending on the sign of $\frac{\partial E}{\partial W}$:

$$V_S = \begin{cases} V_{dd}, & I_{\Delta W} > 0 \\ 0, & I_{\Delta W} < 0 \end{cases}$$  \hspace{1cm} (14.15)

$V_S$ codes the sign of the weight update and it is in input to the local learning rate adaptation circuit $H$.

**The Weight Unit $WU$.** The weight unit $WU$ implements the weight update operations, $W_{k,j}^{new} = W_{k,j}^{old} + \Delta W_{k,j}$, and short-term memory storage of the weight value. Since the weight update signal is coded by a current and the weight by a voltage, the weight update operation is implemented by charging/discharging the weight capacitor [2]. The weight block $WU$ is shown in Figure 14.7 (a): it is composed of an analog switch, implemented by a transfer gate controlled by the update signal $V_u$, and a weight capacitor $C_w$ implemented by NMOS and PMOS transistors in Figure 14.7 (b), using a single poly CMOS process (ATMEL ES2 ECPD07).

![Figure 14.7](image-url)
When the analog switch is closed at time $t_0$ for a time duration $T$, the weight voltage $V_w$ across the weight capacitor becomes:

$$V_w(t_0 + T) = V_w(t_0) + \frac{T}{C_w} \cdot I_{\Delta w} \quad (14.16)$$

The update operation is affected by non-idealities, such as charge injection and charge sharing [7] that modify the final (ideal) value of the weight voltage $V_w(t_0 + T)$. The effects of these non-idealities on the learning process have been discussed in [7]. The Weight Unit implements then a short-term dynamic memory [2]. When the analog switch is opened, the voltage $V_w$ remains stored across the capacitor $C_w$, but leakage currents [39], due to the MOS transistors implementing the switch, charge/discharge the capacitor in a short time, e.g., hundreds of microseconds [39]. During the training phase, the analog switch remains open between two consecutive update operations. This time corresponds to the time necessary to compute the weight update current, a few microseconds. Thus, during the training phase we can neglect the effects of leakage currents on the weight voltage. When the training phase of the network is completed, i.e., the network has learned the task, it performs only the recall phase. Long term storage must be then implemented to keep the weight values at constant values. This can be easily implemented by considering analog non-volatile memory cells, e.g., analog flash memories [29].

The local learning rate adaptation circuit. This unit implements the local adaptation of the learning rate values according to Equation (14.8). The input of the block is $S_{ij}$, coded by the digital value $V_{Sn}$; the output is the current $I_n$. The local learning rate adaptation circuit $H$ is shown in Figure 14.8 [8]. It consists basically of a two-stage dynamic shift register (inverters $I_0$, $I_1$, and $I_2$), an XOR gate ($G_0$), the switches $T_1$ and $T_2$, two capacitors $C_1$ and $C_2$ and some synchronization switches controlled by signals $\varphi_n$, $n = 1, \ldots, 4$.

The transistor $M_1$ works in weak inversion and converts the learning rate control voltage $V_n$, dynamically stored on $C_2$, into the current $I_n$, the bias current of the $B2$ multiplier (Figure 14.6). The voltages $V_H$ and $V_L$ are used to set the minimum and maximum values of the learning rate range, $\eta_{\text{max}}$ and $\eta_{\text{min}}$. The circuit is operated with a four-phase clock scheme through four non-overlapping control signals $\varphi_n$, $n = 1, \ldots, 4$, as shown in Figure 14.9, which also shows the waveform of the signal controlling the weight update switch in Figure 14.7.

The sequence of the operations for one weight update in the $t^{th}$ learning iteration, referring to the logic signal $\bar{S}$ instead of the voltage $V_{Sn}$, is the following:

1. at the beginning all the $\varphi_n$ are low: $B2$ has computed $I_{\Delta w}(t)$ and $S(t)$; $S(t)$ is already stored at the input of inverter $I_0$, and $\bar{S}(t-1)$, computed during the $(t-1)^{th}$ learning iteration, is stored at the input of inverter $I_1$;

2. $\varphi_1$ high; $\bar{S}(t-1)$ is stored at input $B$ of $G_0$;

3. $\varphi_2$ high; $S(t)$ is stored at input $A$ of $G_0$. If $S(t)$ and $\bar{S}(t-1)$ have equal values, node $C$ in Figure 14.8 is connected to the voltage $V_H$, by closing switch $T_1$, otherwise node $C$ is connected to the voltage $V_L$ by closing switch $T_2$;
4. \( \varphi_3 \) high: the value of \( V_{C1}(t) \) is set as follows:

\[
V_{C1}(t) = \begin{cases} 
V_H, & S(t) = S(t - 1) \\
V_L, & \text{otherwise}
\end{cases}
\]  

(14.17)

5. \( \varphi_4 \) high: the capacitors \( C_1 \) and \( C_2 \), where \( C_2 \) stores the voltage \( V_q(t) \), perform charge sharing. The updated value of the learning rate control voltage \( V_q(t + 1) \)
is given by the following expression:

\[ V_n(t+1) = \frac{C_2}{C_1 + C_2} V_n(t) + \frac{C_1}{C_1 + C_2} V_{C_1}(t) \]

\[ V_n(t) + \frac{C_1}{C_1 + C_2} \cdot (V_{C_1}(t) - V_n(t)) \]

\[ V_n(t) + \gamma \cdot (V_{C_1}(t) - V_n(t)) \]  

(14.18)

where \( \gamma = \frac{C_2}{C_1 + C_2} \). With transistor \( M_n \) biased in weak inversion, the new value of the learning rate current \( I_n(t+1) \) is given by:

\[ I_n(t+1) = I_S \cdot e^{-\frac{V_n(t+1)}{2 U_T}} \]  

(14.19)

where \( I_S \) is the specific current of the transistor \( M_n \) [39]. Combining (14.18) and (14.19), we obtain:

\[ I_n(t+1) = I_n(t) \cdot \left( \frac{I_{C_1}(t)}{I_n(t)} \right)^\gamma \]

\[ I_{C_1}(t) = I_S \cdot e^{\frac{V_{C_1}(t)}{2 U_T}} \]  

(14.20)

which implements the learning rate adaptation rule as given in (14.8), where \( \gamma = C_1/(C_1 + C_2) \), and \( I_{C_1}(t) \) corresponds to \( \eta^{\text{max}} \) or \( \eta^{\text{min}} \) according to (14.17). The maximum and minimum values of the learning rate current are then:

\[ I^{\text{max}}_\eta = I_S \cdot e^{\frac{\gamma}{2 U_T}} \]

\[ I^{\text{min}}_\eta = I_S \cdot e^{-\frac{\gamma}{2 U_T}} \]  

(14.21)

For instance, if \( V_H - V_L = 0.3 V \), the ratio between the maximum and minimum values of the learning rate current can be one thousand.

### 14.4.2 The neuron circuit

**The activation function A circuit.** The activation unit \( A \) implements the neuron activation function \( \Psi(a) = \tanh(a) \). The input of the \( A \) block is the current \( I_a \), while the output is the voltage \( V_x \). The \( A \) block must be designed providing the possibility of changing the activation function gain (the slope of the characteristic for zero input) and the saturation values of the transfer characteristic. The circuit implementing the activation function module is shown in Figure 14.10. It is composed of an OTA and two voltage controlled resistors implemented by transistors \( M_a, M_b, M_c, \) and \( M_d \) [38].

The activation current \( I_a \) is converted into the voltage \( V_{in} \) by a resistor \( R_{in} \) implemented by transistors \( M_a \) and \( M_b \) operated in the triode region [39], [38]. The resistor value is controlled by the voltages \( V_p \) and \( V_n \). If transistors \( M_1 \) and \( M_2 \) are biased in weak inversion, we can write:

\[ I_{out} = I_b \cdot \tanh \left( \frac{V_{in} - V_T}{2 n U_T} \right) = I_b \cdot \tanh \left( \frac{R_{in} \cdot I_a - V_r}{2 n U_T} \right) \]  

(14.22)
The current $I_{out}$ is converted into the voltage $V_X$ by a resistor $R_{out}$ implemented by transistors $M_c$ and $M_d$ operated in the triode region; the resistor value is controlled by the voltages $V_{p1}$ and $V_{n1}$:

$$V_X = R_{out} \cdot I_{out} = R_{out} \cdot I_b \cdot \tanh \left( \frac{R_{in} \cdot I_a - V_r}{2nU_T} \right)$$  \hspace{1cm} (14.23)

By changing the values of $R_{in}$ and $R_{out}$ it is possible to change the activation function gain and the saturation values: $R_{in}$ controls the value of the slope for zero input current in (14.23), and $R_{out}$ controls the maximum and minimum values of the output voltage $V_X$.

**The derivative circuit $D$.** The block $D$ computes the derivative of the neuron activation function, $D_k(j) = 1 - (X_k(j))^2$. The input is the voltage $V_X$, and the output is the current $I_d$. The $D$ circuit is implemented using a Gilbert multiplier [27] as shown in Figure 14.11.

The I/O transfer characteristic is:

$$I_d = I_b \cdot \tanh \left( \frac{V_{cem} - V_X}{2nU_T} \right) \cdot \tanh \left( \frac{V_X - V_{sem}}{2nU_T} \right)$$  \hspace{1cm} (14.24)

Since $|V_{cem} - V_X|$ and $|V_X - V_{sem}| \leq 100mV$,

$$I_d = \frac{I_b}{4n^2U_T^2} \cdot (V_{cem} - V_X) \cdot (V_X - V_{sem}).$$  \hspace{1cm} (14.25)

With the values of voltages $V_{cem}$ and $V_{sem}$ such that $V_{cem} - V_r = V_r - V_{sem} = V_{norm}$, Equation (14.25) can be rewritten as:

$$I_d \approx \frac{I_b \cdot V_{norm}}{4n^2U_T^2} \cdot \left( 1 - \left( \frac{V_X - V_r}{V_{norm}} \right)^2 \right).$$  \hspace{1cm} (14.26)
The error circuit $R$. The error circuit $R$ computes the error terms in the output and hidden neurons, $\delta_k = (X_k - X_k) \cdot D_k$ and $\delta_i = \left( \sum_{j=1}^{n} \delta_j \cdot W_{j,i} \right) \cdot D_i$ respectively. In the output layer the inputs of this block are the voltages $V_T$ and $V_X$, while inputs to the hidden layer inputs are voltages $V_{sw}$ and $V_r$. The output is the voltage $V_5$. The error circuit $R$ is shown in Figure 14.12.
ON-CHIP LEARNING RATE ADAPATION

It is composed by an OTA circuit [27] and a resistor $R_{out}$, implemented by transistors $M_c$ and $M_d$, and controlled by voltages $V_{p1}$ and $V_{n1}$, that converts the OTA output current $I_{out}$ into the error voltage $V_6$. The I/O transfer characteristic is:

$$V_6 = R_{out} \cdot I_d \cdot \tanh \left( \frac{V_1 - V_2}{2nU_T} \right)$$

(14.27)

If $|V_1 - V_2| \leq 100mV$,

$$V_6 \approx \frac{R_{out} \cdot I_d}{2nU_T} \cdot (V_1 - V_2)$$

(14.28)

In the output layer $V_1 = V_T$, $V_2 = V_X$, and in the hidden neurons $V_1 = V_{out}$, and $V_2 = V_T$ (see Figure 14.4 and Table 14.1).

The error cost function circuit $FC$. The $FC$ block computes the quadratic error between the target and the corresponding output of the output neuron, $e = (X_k - X_k')^2$. The inputs of the $FC$ circuit are mapped onto voltages. The output is mapped onto the current $I_r$. The block $FC$ is implemented by using a Gilbert multiplier where the differential input voltages of each OTAs are equal to $V_T - V_X$.

14.4.3 Physical design and implementation of the analog architecture

A prototype chip implementing the on-chip neural architecture with local learning rate adaptation discussed in Section 14.3 has been designed and fabricated through the ATMEL ES2 ECPD07 CMOS technology available through the Europractice Service. This is a digital single-poly, double metal CMOS process with 0.7µm minimum channel length. The synaptic and neuron modules have been designed according to the structure discussed in Section 14.3 and the circuit implementation discussed in Section 14.4. The prototype chip implements a MLP network with 8 inputs, 16 hidden neurons, and 4 output neurons; each neuron is provided an extra synapse implementing the bias unit [32, 19]. Globally the chip implements 212 synaptic modules and 20 neuron modules. The chip has the 8 MLP inputs and 4 the MLP outputs directly accessible, and the outputs of the 16 hidden neurons are multiplexed onto a single signal. The chip micrograph and layout floorplan are shown in Figure 14.13.

S1 and N1 are the synaptic matrix and the neuron array related to the first (hidden) layer, and S2 and N2 are the synaptic matrix and the neuron modules array related to the second (output) layer. The block B contains the circuits needed to generate and distribute the bias voltages and currents to all the circuits on the chip. The block CU contains some digital logic needed to address and access the weight voltages of the synaptic modules. The chip features an area of about $3.5mm \times 3.5mm$ and contains about 22,000 transistors.

14.5 EXPERIMENTAL RESULTS

Figure 14.14 shows the I/O transfer characteristic of the $W$ block of Figure 14.5. The quadratic behavior is due to transistors $M_1$, $M_2$, $M_3$, and $M_4$ biased in strong inversion.
We measured the behaviour of a single perceptron in Figure 14.15 comprising the
A block of a hidden neuron (the \( j \)th hidden neuron) and 8 \( F \) multipliers connecting
the 8 MLP inputs to the neuron.

The inputs of the perceptron are the voltages \( V_i^j \), \( i = 1, \ldots, 8 \). The weight of each
\( F \) multiplier is set by voltages \( V_n^j \). The outputs of each multiplier, the currents \( I_{nX}^j \),
are summed and presented to the input of block A that applies the activation function
and gives as output the voltage \( V_X^j \). According to the circuit description discussed in
Section 14.4:

\[
V^j_X = R_{out} \cdot I_b \cdot \text{tanh} \left( \frac{R_in \cdot \left[ \sum_i g \left( V^{j,i}_{W,i} \right) \cdot \frac{(V^j_X - V_r)}{2nU_T} \right] - V_r \right)
\]

(14.29)

The synaptic input voltages \( V^{j,i}_X, i = 2, \ldots, 8 \) were set to \( V_r \), the signal ground. Consequently, the output currents of the corresponding multipliers \( I^{j,i}_{W,i}, i = 2, \ldots, 8 \) were (ideally) null. Equation (14.29) then becomes:

\[
V^j_X = R_{out} \cdot I_b \cdot \text{tanh} \left( \frac{R_in \cdot g \left( V^{j,1}_{W} \right) \cdot \frac{(V^j_X - V_r)}{2nU_T} - V_r \right)
\]

(14.30)

Figure 14.16 shows the I/O transfer characteristics of the single perceptron, where the synaptic input voltage spans the range [2.4V, 2.6V], and the weight voltage varies between 1V and 4V in steps of 1V. The voltage \( V_r \) was set to 2.5V, \( I_b \) to 1\( \mu A \). The values of \( R_{in}, R_{out} \) were set by imposing \( V_n = V_{n1} = 4.5V, \) and \( V_p = V_{p1} = 0.5V \). The offset in the transfer characteristics is mainly due to the zero-offsets of the synaptic multipliers whose input voltage is set to the signal ground, \( V^j_X = V_r = 2.5V, i = 2, \ldots, 8 \).

Figure 14.17 illustrates the transient response of the circuit of Figure 14.15. Figure 14.17 (a) is obtained for a positive weight value, \( V^{j,1}_{W} = 4V \), while Figure 14.17 (b) is obtained for a negative weight value \( V^{j,1}_{W} = 1V \).
Figure 14.16 I/O characteristics of the circuit of Figure 14.15.

Figure 14.17 Transient response of the circuit of Figure 14.15 for positive (a) and negative (b) weight values. Upper traces: synaptic input signals; bottom traces: neuron output signals.

The response time is about 1 μs; consequently, the response time of two cascaded perceptrons, implementing a two-layer MLP, is about 2μs. In Figure 14.18 the transfer characteristics of the FC circuit are shown.

The behaviour of current $I_t$ versus the neuron output voltage $V_X$ is plotted for two different values of target voltage $V_T$, 2.4V and 2.6V respectively. The transfer characteristics follow the desired quadratic profile. Figure 14.19 shows the observed weight decay due to the leakage currents of the switch that charge/discharge the weight capacitor $C_w$ as described in Section 14.4 and Figure 14.7.

We stored a voltage value on a weight capacitor and then we looked at the transient behaviour of the corresponding current $I_W$ with the capacitor floating. Figure 14.19 (a) shows the transient behaviour of the current $I_W$ for different values of the initial
Figure 14.18 Transfer characteristics of the FC circuit.

Figure 14.19 Weight decay due to leakage currents on the weight capacitor $C_w$.

weight voltages. Leakage currents charge/discharge the weight capacitor and the corresponding current $I_w$ approach to zero. In Figure 14.19 (b) the values of the current $I_w$ measured after 60 seconds versus the corresponding initial values are
Figure 14.20 Training phase of the MLP chip. (a) Top trace: target signal; bottom trace: output signal. (b) Single weight behaviour during training.

shown. After 60s the weight values have degraded by about 6%, i.e., the precision is about 4 bits. We can expect 8 bit precision by refreshing the weight voltages every 2 seconds.

Figure 14.20 illustrates the transient behaviour of the MLP chip during a training phase of the network learning a two-input XOR function. We configured the network as a $2 \times 2 \times 1$ MLP, where the unused input and hidden units were turned off by biasing them at the signal ground, $V_r$ (2.5V). For training the two-input XOR function, we use the local learning rate adaptation circuits, where the minimum and maximum learning rate control voltages, $V_L$ and $V_H$, were set to 0.4V and 0.6V respectively. The learning iteration time duration was set to 800µs. In Figure 14.20 (a) the target signal (top trace) and the output signal (bottom trace) at the end of the training phase are shown: the network learn the two input XOR function after about 40 learning phases. In Figure 14.20 (b) the behaviour of a MLP weight signal during the training phase is shown: after about 32 ms, or 40 learning iterations, this weight reached its final value and the learning was completed.

14.6 CONCLUSIONS

We implemented a Multi-Layer Perceptron neural network with on-chip by-pattern Back Propagation learning. To improve the training properties, we proposed and implemented a local learning rate adaptation technique, which drastically improves the convergence speed. Important considerations in favor of this learning rate adaptation technique are the following: (i) it is local, in that each synapse has its own learning rate adapted according to local properties; (ii) it is suited for analog VLSI implementation; and (iii) the adaptation rule can be extended to many other learning algorithms to train feedforward and recurrent neural networks, including model-free techniques. The neural paradigm and the resulting learning rule have been validated on the recognition of hand-written digits. The MLP neural paradigm and the on-chip learning algorithm has been mapped onto an analog circuit architecture. A prototype chip implementing the
Table 14.2 Chip characteristics

<table>
<thead>
<tr>
<th>Network size</th>
<th>8 x 16 x 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-chip learning algorithm</td>
<td>by-pattern BP with local learning rate adaptation</td>
</tr>
<tr>
<td>Technology</td>
<td>ATMEL ECPD07</td>
</tr>
<tr>
<td>Transistor count</td>
<td>22000</td>
</tr>
<tr>
<td>Chip size</td>
<td>3.5 x 3.5mm²</td>
</tr>
<tr>
<td>Computational power</td>
<td>266KCUPS</td>
</tr>
<tr>
<td>Computational density</td>
<td>21.6KCUPS/mm²</td>
</tr>
<tr>
<td>Energy efficiency</td>
<td>10.6KCUPS/mW</td>
</tr>
</tbody>
</table>

on-chip neural architecture with local learning rate adaptation has been designed and fabricated in ATMEL ECPD07 CMOS technology, a digital single-poly, double-metal CMOS process with 0.7 µm minimum channel length. The chip implements an MLP neural network with 8 inputs, 16 hidden neurons, and 4 output neurons. The chip implements 212 synaptic and 20 neuron modules, occupies an area of 3.5 × 3.5mm², and consumes about 25 mW. Learning experimental results confirm the chip functionality, featuring a computational power of 265 KCUPS, a computational density of 21.6 KCUPS/mm², and an energy efficiency of 10.6 KCUPS/mW. The main features of the chip are reported in Table 14.2.

Notes

1. Note that sometimes the objective of learning is to adapt slowly to a changing environment, so slow learning is sometimes an advantage rather than a disadvantage. This chapter addresses applications where fast learning is desirable.

2. Assuming the voltage signals are sufficiently low-impedance to drive the input impedance of the elements to which they are distributed.

3. V denotes voltage signals and I denotes current signals; the indices i, j, and k are not indicated for the sake of simplicity.

4. Thus, a voltage equal to 2.5V corresponds to a null neural variable.

5. g_{w}(V_w) is not explicated for the sake of simplicity.

6. e.g., V_{cm}=2.6V, V_{sam}=2.4V, V_r=2.5V, and then V_{norm}=0.1V.

References


ON-CHIP LEARNING RATE ADAPTATION


