Fully Parallel Summation in a New Stochastic Neural Network Architecture.

C.L. Janer, J.M. Quero and L.G. Franquelo, Member, IEEE
Escuela Superior de Ingenieros
Avda. Reina Mercedes s/n
Sevilla 41012 SPAIN

Abstract—A space efficient fully parallel stochastic architecture is described in this paper. This stochastic architecture circumvents the main drawback of stochastic implementations of neural networks — the concurrent processing of a high number of weighed input signals, leading to a simple realization of stochastic summation. An unlimited number of stochastically coded pulse sequences can be added in parallel using only very simple and space efficient digital circuitry. Any neural network, either recurrent or feedforward, can be implemented using this scheme provided that neurons take discrete values. Design criteria are deduced from the mathematical analysys of the involved stochastic operations. Simulation results are also given.

I. Introduction

Electronic realization of neural networks can be faced in different ways. On one hand analog approaches are very simple in terms of circuitry and have fast convergence times, specially when they are compared with digital implementations, but on the other hand their programming flexibility is very low. Digital implementations perform high flexibility and easy interface with general purpose computers but their efficiency in terms of consumed silicon area is very low, as a floating point multiplier is needed in every neuron to calculate the presynaptic activity. One way to circumvent this problem is to employ stochastic logic[1].

Stochastic logic systems realize pseudoanalog operations using stochastically coded pulse sequences. Multiplication of two stochastic pulse sequences should produce another stochastic stream of pulses whose firing probability is the product of the input firing probabilities. This can be achieved easily if the input sequences are stochastically independent. The circuit that implements this operation is a simple AND gate.

Stochastic summation is a much more difficult operation to perform, specially if the terms to be added are signed. Two types of circuits have been described in the bibliography. One is the OR gate [2] and the other is an up/down counter[3]. If two pulse sequences are feeded into an OR gate and the pulse sequences to be added do not overlap, the output firing probability is equal to the sum of both firing probabilities This OR-based add function is thus distorted by pulse overlap. In order to achieve a quasy linear behaviour pulse densities should stay very low, specially if many terms are to be added. This technique does not permit the integration of neurons with a very high number of synaptic connections as it would lead to extremely low maximum pulse density.

The up/down counters technique, although is widely used [4], has a very important drawback. The pulses coming from other neurons have to be multiplexed in time (i.e. *serialized*) leading to *high computation times* if the network has many neurons and many synapsys per neuron.

We propose a fully parallel stochastic architecture for neural networks whose neuron activity values take discrete values (either $-1, 1$ or $0, 1$). This architecture permits the integration of highly interconnected neural networks. It can be used either for recursive or feedforward nets and it is very efficient in terms of circuitry.

This paper is organized as follows. In section 2 the accuracy of stochastic multiplication is analyzed. The obtained results justify the scheme proposed in section 3, where design criteria are given. Section 4 is devoted to some applications of this novel architecture. Finally in section 5 conclusions are drawn.

II. Stochastic Multiplication.

Fig.1 shows the generation and multiplication of a set of n independent stochastic signals. The output value obtained by accumulating the pulses N clock-cycles follows a binomial distribution having the expectation value E and the variance V.

$$E[\tilde{x}] = \mu = Nx \tag{1}$$

$$V[\tilde{x}] = \sigma^2 = Nx(1-x) \tag{2}$$

with

$$x = \frac{\prod_{i=1}^{i=n} a_i}{a_{max}^n} \tag{3}$$

where a_i is the stored value and a_{max} is the maximum random number that can be generated. A natural way to evaluate the rate error of the

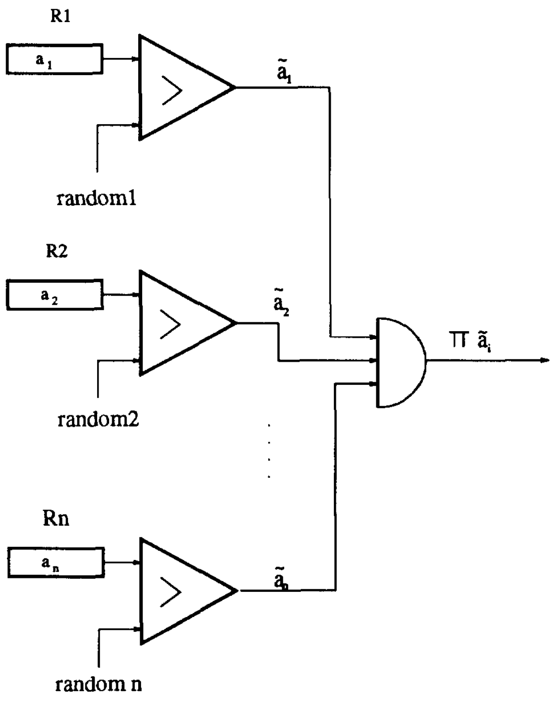

Figure 1: Stochastic multiplier.

stochastical multiplication result is to consider the following fraction

$$\frac{\sigma}{\mu} = \frac{\sqrt{N}\sqrt{x}\sqrt{1-x}}{Nx} \tag{4}$$

The smaller this fraction is, the more accurate the aproximation will be. This expression is a function of the ideal product x and the number of generated random values N. It has been plotted in Fig.2 and Fig. 3. Notice that if N is big enough $\frac{\sigma}{\mu}$ only depends slightly on x for x taking values not too close to zero. This fact is of most importance as it will become clear in next section.

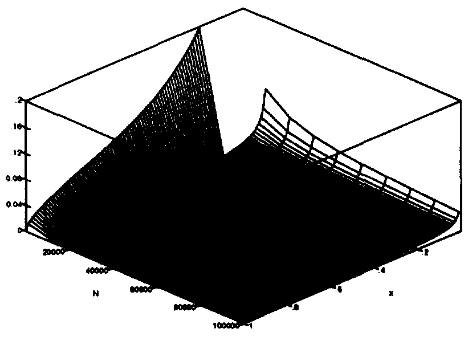

Figure 2: Product rate error.

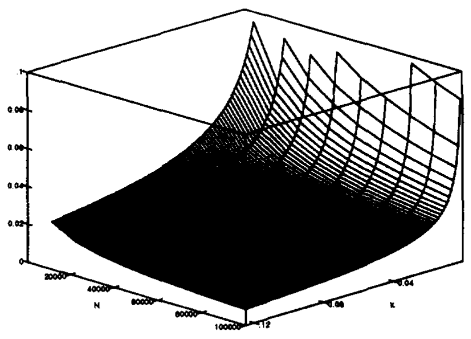

Figure 3: Product rate error (x closed to 0).

III. Stochastic Architecture

A. Basic Concepts.

The transfer function of a discrete neuron requires the application of the sign operator to the summation of weighed input signals. Consider the following identity

$$sign(\sum_{i=1}^{i=n} x_i) = sign(\sum_{i=1}^{i=n} x_i^+ - \sum_{i=1}^{i=n} x_i^-) =$$

$$sign(e^{-\sum_{i=1}^{i=n} x_i^-} - e^{-\sum_{i=1}^{i=n} x_i^+}) =$$

$$sign(\prod_{i=1}^{i=n} e^{-x_i^-} - \prod_{i=1}^{i=n} e^{-x_i^+}) \tag{5}$$

where

$$x_i^+ = \begin{cases} x_i & \text{for } x_i > 0 \\ 0 & \text{for } x_i < 0 \end{cases} \tag{6}$$

and

$$x_i^- = \begin{cases} -x_i & \text{for } x_i < 0 \\ 0 & \text{for } x_i > 0 \end{cases} \tag{7}$$

The last term in (5) can be regarded as the comparison of two pulse streams generated by two stochastic multipliers, therefore no adder is needed.

If the neural network has been adimentionalized in such a way that all terms to be aggregated take values ranging from zero to a small number close enough to zero, e^{x_i} can be aproximated by

$$e^{-x_i} \simeq 1 - x \tag{8}$$

In Fig.4 rate error $((e^{-x} - (1-x))/e^{-x})$ is plotted as a function of the number to be transformed x. This transformation is carried out because the accuracy of stochastic multiplication does not decrease strongly as x becomes closed to zero. If this did happen N would have to be large leading to slow dynamics.

B.Functional Description.

The fully parallel stochastic architecture is shown in Fig.5. In block M synapsys are compared with random numbers producing a set of uncorrelated streams of pulses whose densities are proportional to their values. The evaluation of (8) is performed by a simple NOT gate (block E). Block S is a logic block, where pulses are separated in either "positive" if weight and neuron values were equaly signed or "negative" if they were not. "Positive" and "negative" streams are multiplied separately, leading to two diferent stochastic signals. Two different implementations are suggested for neurons with either $\{-1, 1\}$ or $\{0, 1\}$ saturation states, as shown in Fig.6 and 7 respectively, where

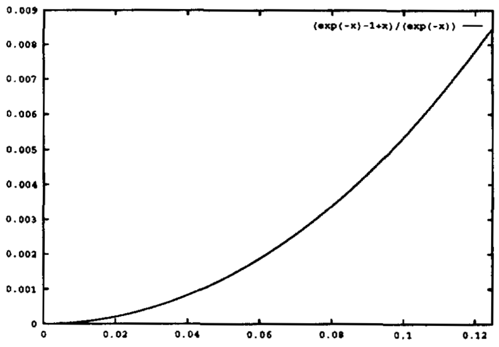

Figure 4: Multiplication rate error.

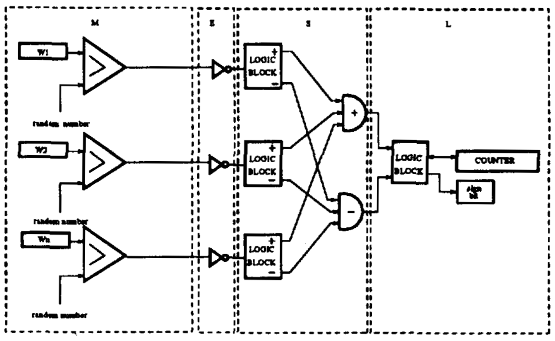

Figure 5: Architecture.

negative sign is coded by "1." In block L if the two signals are at high level at the same time the up/down counter remains unchanged. If only a "positive" pulse is at high level the counter is incremented by one and if the pulse is "negative" the counter is then decremented by one. Block L resets the sign bit if a zero crossing takes place.

Positive and negative terms are transformed and then are multiplied separately as it is shown in Fig.5. Due to the fact that these terms are close to unity many terms can be aggregated with a high degree of accuracy as it becomes clear from Fig.2 and Fig.3.

C. Maximun Error Boundary.

We shall denote the errors associated to the product term calculation x_i by ϵ_i, the errors associated to the exponential transformation by ξ_i and the error associated to aggregation by ζ

$$\begin{array}{ccccc} x_1+\epsilon_1 & & e^{x_1+\epsilon_1}+\xi_1 & & \\ x_2+\epsilon_2 & & e^{x_2+\epsilon_2}+\xi_2 & & \\ \cdot & \longrightarrow & \cdot & \longrightarrow & \\ \cdot & & \cdot & & \\ \cdot & & \cdot & & \\ x_n+\epsilon_n & & e^{x_n+\epsilon_n}+\xi_n & & \\ \prod_{i=1}^{i=n}(e^{x_i+\epsilon_i}+\xi_i)+\zeta & & & & \end{array}$$

Taking into account that errors are small and denoting rate errors by $\hat{\epsilon}_i, \hat{\xi}_i, \hat{\zeta}$ we obtain

$$\prod_{i=1}^{i=n}(e^{x_i+\epsilon_i}+\xi_i)+\zeta \simeq \prod_{i=1}^{i=n} e^{x_i+\epsilon_i}+\sum_{i=1}^{i=n}\xi_i\prod_{j\neq i} e^{x_j+\epsilon_j}+\zeta =$$

$$\prod_{i=1}^{i=n} e^{x_i(1+\hat{\epsilon}_i)}(1+\sum_{j=1}^{j=n}\hat{\xi}_j+\hat{\zeta}) \qquad (9)$$

If we evaluate the logarithm of this expression and define $\hat{\hat{\xi}}_i := \frac{\hat{\xi}_i}{x_i}$ we have

$$\sum_{i=1}^{i=n} x_i(1+\hat{\epsilon}_i)+Log(1+\sum_{j=1}^{j=n} x_j\hat{\hat{\xi}}_j+\hat{\zeta}) \qquad (10)$$

Substracting $\sum_{i=1}^{i=n} x_i$ we obtain the total error

$$err \leq \sum_{i=1}^{i=n} x_i\hat{\epsilon}_m + Log(1+\sum_{j=1}^{j=n} x_j\widehat{\hat{\xi}_m}+\hat{\zeta}) \leq$$

$$\sum_{i=1}^{i=n} x_i\hat{\epsilon}_m + \sum_{j=1}^{j=n} x_j\widehat{\hat{\xi}_m}+\hat{\zeta} \qquad (11)$$

the expression of the rate error is

$$\hat{err} = \frac{err}{\sum_{i=1}^{i=n} x_i} \leq \hat{\epsilon}_m + \widehat{\hat{\xi}_m}+\hat{\zeta} \qquad (12)$$

Errors $\hat{\epsilon}_m$ and $\hat{\zeta}$ tend to zero when N increases, and $\widehat{\hat{\xi}_m}$ can be done small enough if x is chosen within a proper range, as shown in Fig.8. For instance, if the pulse stream is considered to have a mean value of 0.12, its exponential transformation error will remain bellow 7%.

IV. Simulation Results

The dynamic behaviour of a discrete valued Hopfield neural network has been simulated. Three 25 neuron patterns, showed in Fig.9a,c were stored in this network.

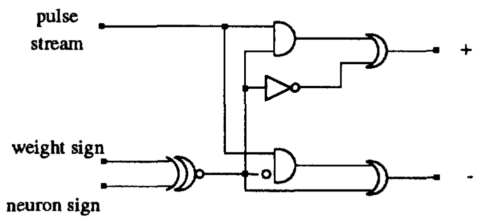

Figure 6: Logic block in S for neurons with saturated states $\{-1,1\}$.

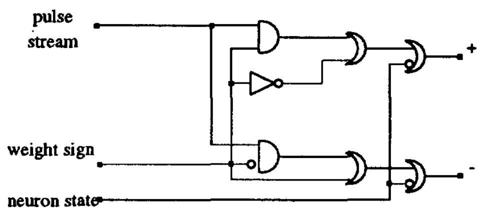

Figure 7: Logic block in S for neurons with saturated states $\{0,1\}$.

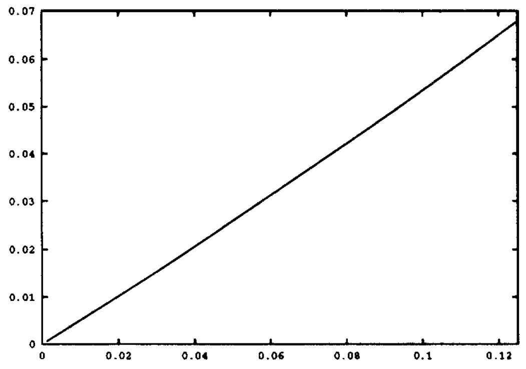

Figure 8: Transformation rate error.

$$\begin{matrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1\\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1\\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1\\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1\\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1\\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1\\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1
\end{matrix}$$

(a)

$$\begin{matrix}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1
\end{matrix}$$

(b)

$$\begin{matrix}
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{matrix}$$

(c)

$$\begin{matrix}
1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1\\
1 & 1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 & 1\\
1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1\\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1\\
1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 & 1\\
1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & 1\\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & -1\\
1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1\\
1 & 1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 & 1\\
1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1
\end{matrix}$$

(d)

Figure 9: Patterns stored in a hopfield net.

Fig.10 shows the evolution of the Hamming distance between the neural state vector and the prototype vector (Fig.9a) for different a_{max}. They are also compared with the evolution of the architecture of Van den Bout [3]. The initial state was a corrupted vector of data (Fig.9d) resembling this stored pattern. For the suggested architecture, the number of clock cycles needed for full convergency is 25 when a_{max} = 0.0625, and 74 when a_{max} = 0.125. If a systolic array of parallel neuron processors [3] is used the evolution is slower, as showed in Fig.10, needing 100 clock cycles. The number of connections summed up in each clock cycle is N in every neuron while only one is possible in the systolic array. In order to test the behavior of this architecture in feedforward networks the two layer perceptron needed in [5] to carry out nondestructive evaluations has been implemented. The aim of this network is to classificate a set of input signals into four categories. Ten hidden units with eight input signals and two output units were configured using backpropagation

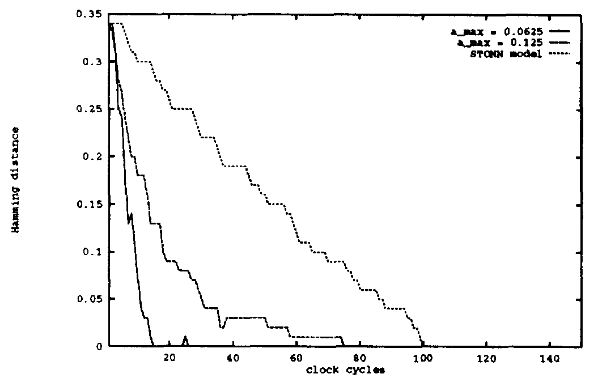

Figure 10: Dynamic behaviour of a hopfield net.

[6]. **Fig.11 represents the dynamic behaviour of its two output neurons when applying one of the previously learnt patterns. In the steady state, the output counters reach limit counts of** -128 **and** 128**, corresponding to output neurons states** -1 **and** 1 **respectively, which are the associated target output values of the applied pattern. The pseudorandom evolution of the values of the output neurons are defined by the evolution of the neurons in the hidden layer. All patterns were applied, leading to their corresponding output vector.**

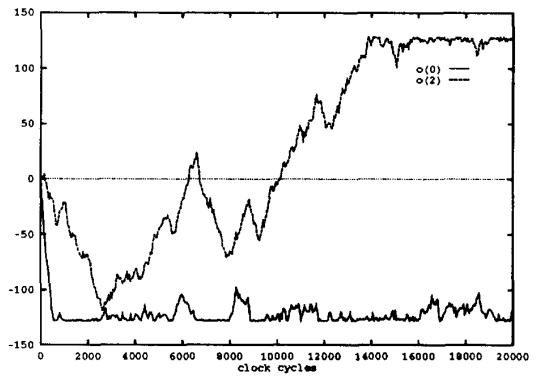

Figure 11: Dynamic behaviour of a two-output one-hidden layer perceptron.

V. Conclusions

A new approach to summation of stochastic weighed signals has been presented. The number of concurrent input signals is no longer a limit. The evaluation of these signals is carried out in every cycle, leading to a full parallel implementation. Limiting the range the input signals allows for a very simple implementation. Simulation results validates this model for recurrent and feedforward networks.

References

[1] **Y. Kondo and Y. Sawada. Functional Abilities of a Stochastic Logic Neural Networks** *IEEE Trans. on Neural Networks*, **vol.3, pp.434-443, 1992.**

[2] **Alan f. Murray, Dante Del Corso and Lionel Tarassenko. Pulse-Stream VLSI Neural Networks Mixing Analog and Digital Techniques** *IEEE Trans. on Neural Networks*, **vol.2,no.2, pp.193-204, 1991.**

[3] **D.E. Van den Bout and T.K. Miller III. A Digital Architecture Employing Stochaticism for the Simulation of Hopfield Neural Nets.** *IEEE Trans. on Circuit and Systems*, **vol.36, pp. 732-738. 1989**

[4] **W. Wike, D.E. Van den Bout and T.K. Miller III The VLSI Implementation of STONN.** *IEEE Int. Joint Conf. on Neural Networks*, **vol.2, pp.593-598, 1990.**

[5] **L. Udpa and S. S. Udpa. Application of Neural Networks to Nondestructive Evaluation.** *First IEE Conference on Artificial Neural Networks*, **pp. 143-147. 1989**

[6] **D. E. Rumelhart and J. L. McClelland. Parallel Distributed Processing. MIT Press. 1986.**