pp. 368-377

Microelectronics for dynamic neural networks

Stefan J. PRANGE *
Axel JAHNKE *
Heinrich KLAR *

Abstract

Artificial Neural Networks *are the massively parallel interconnection of simple processing elements. Computing times for the simulation of these parallel systems on today's von-Neumann-computers increase with the squared number of processing elements. There is a need for application specific hardware. This paper describes various investigations of analog as well as digital hardware for neural networks. Possible solutions for the connection problem and different circuit designs will be explained. Then our cascadable digital circuit for the emulation of a biology-oriented, dynamic neural network will be presented.*

Key words : Neural network, Special purpose computer, Interconnexion, Cascade arrangement, Two dimension network, Analog circuit, Digital circuit, Dynamic model, Circuit design.

MICROÉLECTRONIQUE POUR RÉSEAUX NEURONAUX DYNAMIQUES

Résumé

Les réseaux neuronaux artificiels consistent en une interconnexion massivement parallèle de processeurs simples. Les temps de calcul nécessaires à leur simulation sur les ordinateurs actuels du type von Neumann croissent comme le carré du nombre de processeurs. Il faut donc développer du matériel spécialisé. L'article décrit des études de matériels analogiques et numériques pour réseaux neuronaux. Il présente des solutions pour résoudre le problème de connexions et différentes réalisations par des circuits électroniques. Il décrit aussi un circuit numérique pouvant être monté en cascade pour émuler un réseau neuronal dynamique modélisant un système biologique.

Mots clés : Réseau neuronal, Calculateur spécialisé, Interconnexion, Montage cascade, Réseau bidimensionnel, Circuit analogique, Circuit numérique, Modèle dynamique, Conception circuit.

Contents

I. INTRODUCTION

The term *Artificial Neural Networks* stands for the massive parallel interconnection of simple processing elements. Following biology, these processing elements are often called *neurons* or *neuron models*. The connection elements are called *synapses*, respectively. In neural networks mostly used today the synapses are multipliers and the neurons are adders with a subsequent threshold function, e.g. the sigmoid function [13].

The most popular network architecture is the multilayer feedforward architecture (Fig. 1), i.e. the neurons are arranged in layers and the neurons of two subsequent

* Institut für Mikroelektronik Technische Universität Berlin, Jebensstr. 1, D-1000 Berlin 12, Allemagne.

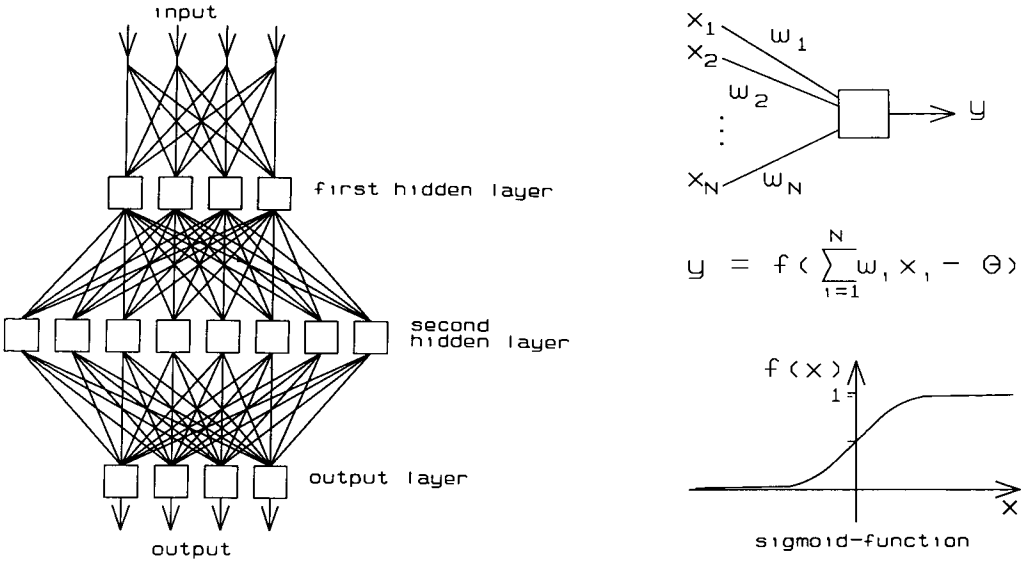


FIG. 1. — Multilayer network architecture and processing element [13].

Réseau à architecture multicouche et processeur élémentaire.

layers are fully interconnected in forward direction via synapses. There is a learning algorithm for this network architecture, the so-called error-back propagation [13]. A learning algorithm is a strategy to iteratively change the synaptic weights (the multiplicands) so that a set of input values is associated with a set of output values as desired.

The computing times for the simulation of neural networks on current digital computers based on the von-Neumann-architecture increase with the number of synaptic interconnections, i.e. the squared number of the neurons in fully interconnected network architectures. These computing times are too long for realistic applications, such as associative image pattern recognition. There is a need for specific hardware [11].

II. THE CONNECTION PROBLEM

One problem often mentioned regarding the hardware realization of neural networks is the so-called *connection problem*, i.e. how can network architectures be mapped onto chip architectures without creating too many wire crossings. In the main there are two strategies to avoid this problem. One possibility especially for general-purpose *neurocomputers* is to combine each neural output value with an address (e.g. [1, 21]). A second possibility is the use of architectural variations. In that case, it has to be taken into account that a considerable area is needed for the circuit design of the synaptic interconnections.

Every kind of network can always be described by a matrix, the entries of which stand for the connections between the nodes. A fully interconnected neural network with N neurons is described by an $N \times N$ matrix (an $\{N \times N \times$ number of parameters$\}$-tensor, respectively) containing the synaptic transmission parameters. The optimal layout for the synapse circuits of a fully parallel, fully interconnected neural network is the form of a square matrix. The input lines can, e.g., be placed horizontally over this matrix, the summation lines vertically. The neuron blocks can be arranged in a row beneath this matrix. Area can be saved and regularity increased by integrating the connections between neural outputs and synaptic inputs into the synapse cells (Fig. 2). By this, an abutment design of cells of the same kind becomes possible.

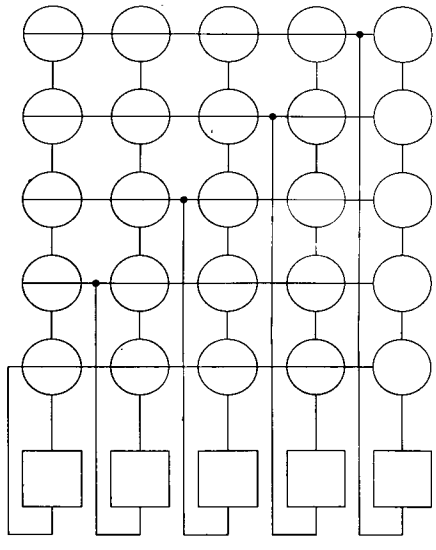


FIG. 2. — Compressed matrix architecture for fully interconnected neural networks.

Architecture matricielle comprimée pour réseaux neuronaux à interconnexion totale.

This matrix architecture can easily be made cascadable by connecting the input- and output-lines as well as the summation lines to pads. If such standard chips are cascaded to a larger neuron matrix- the neuron blocks of some chips remain unused (Fig. 3).

A fully interconnected network architecture can be used for every kind of network architecture by setting the multiplicative parameters of unused synapses to zero.

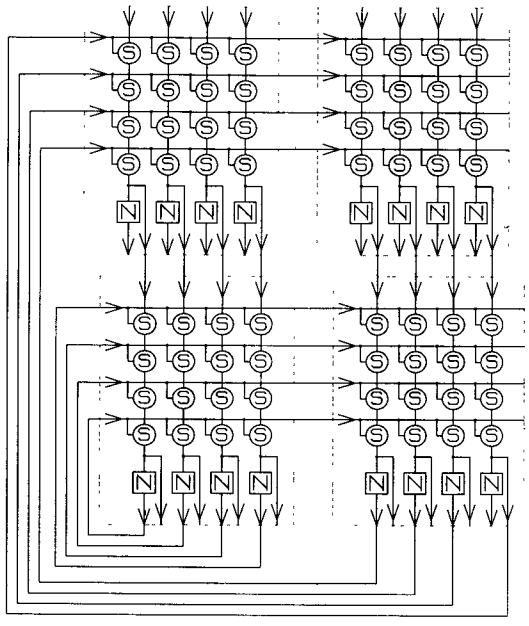


FIG. 3. — Cascaded matrix architecture for fully interconnected networks.

Architecture matricielle en cascade pour réseaux à interconnexion totale.

However, circuit effort can considerably be reduced and, by this, area can be saved if only those synaptic interconnections that are implemented are really needed. E.g., a 3-layer feedforward network with N_1, N_2 and N_3 neurons in the first, second and third layer needs $N_1 \times N_2 + N_2 \times N_3$ instead of $(N_1 + N_2 + N_3)^2$ synaptic interconnections. The optimal architecture for such a network is one $N_1 \times N_2$-matrix and one $N_2 \times N_3$-matrix of synaptic interconnections (Fig. 4). It follows that multilayer networks can easily be built up with the standard chips described, because again matrices are needed (Fig. 5). Even if feedback between layers is desired, the architecture constructed out of matrices is the optimal way. E.g., full feedback between layer 3 and 1 needs an $N_3 \times N_1$ connection matrix.

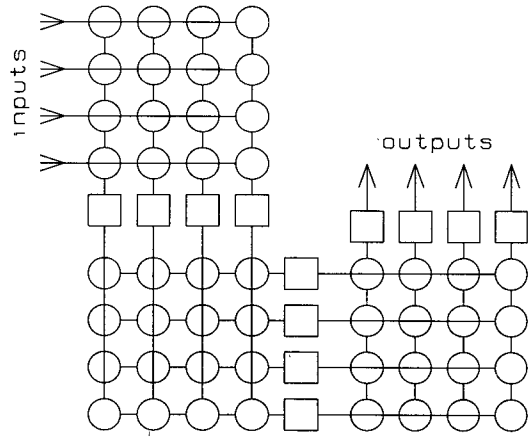


FIG. 4. — Optimized matrix architecture for multilayer feedforward networks.

Architecture matricielle optimisée pour réseaux multicouches à antéroaction.

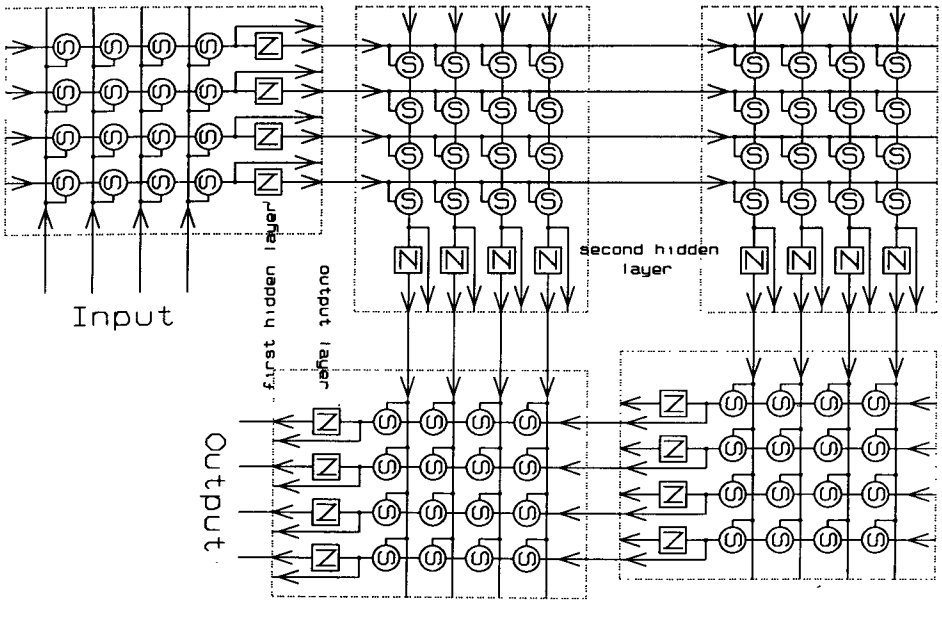


FIG. 5. — Matrix architectures cascaded to a multilayer network.

Architectures matricielles montées en cascade pour constituer un réseau multicouche.

Biology-oriented neural networks often have an architecture where every neuron is connected to a defined number of neighbours. The biological background is that the probability that two cells are connected decreases with their distance [10]. This is also expressed by the Mexican hat function for the synaptic weight parameters, if it is approximated by zero for a certain distance value. That type of architecture is also known from the Kohonen feature map [12] or from inhibitory networks. It will be called neighbourhood network in the following.

The strategy to find the optimal chip architecture for these neighbourhood network architectures is quite simple. It is demonstrated for a simple neighbourhood network, in which each neuron is connected to its nearest neighbours only (Fig. 6). Take a square matrix containing all possible connections and erase every connection that remains unused (Fig. 7). Then try to compress the remaining graph (Fig. 8). The result is rather regular for the one-dimensional case (Fig. 9), more complicated two-dimensional cases (Fig. 10), and even for three-dimensional networks, too. Cascadability is easy for the one-dimensional case if the number of neighbours is defined. In the two-dimensional case regular substructures can be found, too. More flexible cascading strategies for these types of networks are looked for.

The structures shown cannot only be architectures for integrated circuits but also describe the data flow in systolic architectures, in parallel processors or in programs.

Another way to avoid the connection problem is to reduce the connectivity to a local neighbourhood as in the simple neighbourhood network. In the most simple case of so-called cellular neural networks [4] the neurons are connected to their nearest neighbours only. Two-dimensional architectures of this type can directly be mapped onto hardware [7]. However, those networks require a different theoretical approach.

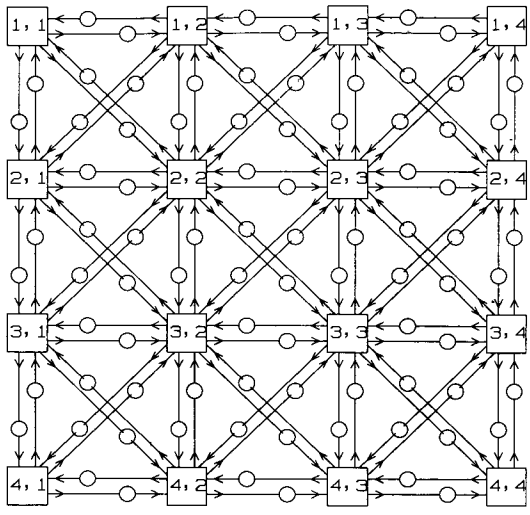


FIG. 6. — Simple two-dimensional neighbourhood network.

Réseau à voisinage bidimensionnel simple.

III. POSSIBILITIES OF ANALOG HARDWARE IMPLEMENTATIONS

After the basic architectural problems have been shown in the previous chapter, this chapter describes starting points for an analog circuit design of neural hardware. As shown above most of the area will be spent for the synapse circuits. The main advantage of an analog circuit design is the possibility to sum up

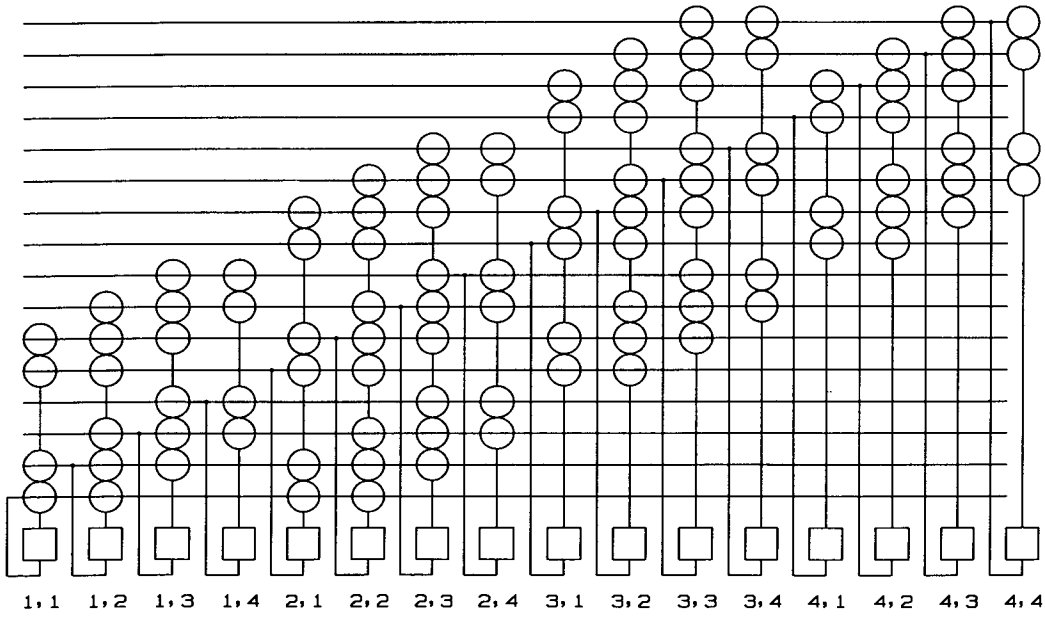


FIG. 7. — Simple two-dimensional neighbourhood network, first step towards an optimized architecture.

Première étape vers une architecture optimisée pour réseau à voisinage bidimensionnel simple.

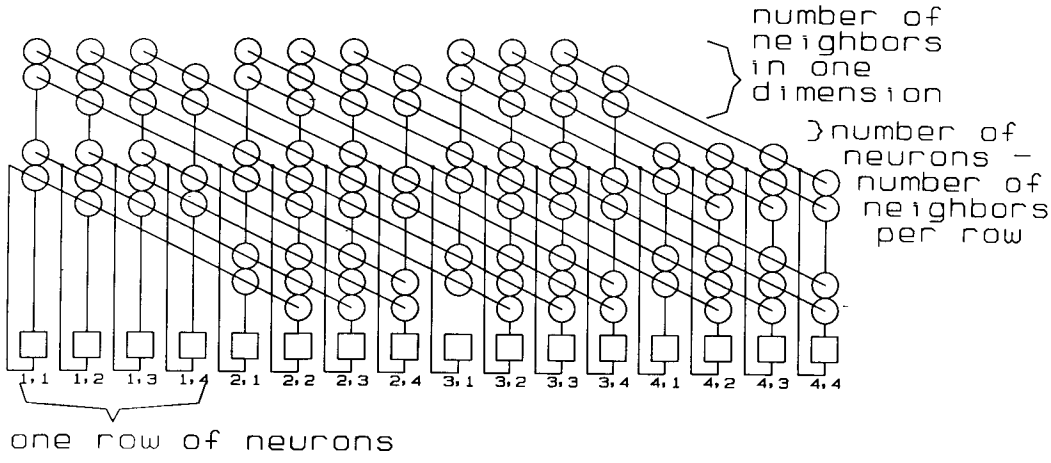


FIG. 8. — Optimized architecture for the simple two-dimensional neighbourhood network.

Architecture optimisée pour réseau à voisinage bidimensionnel simple.

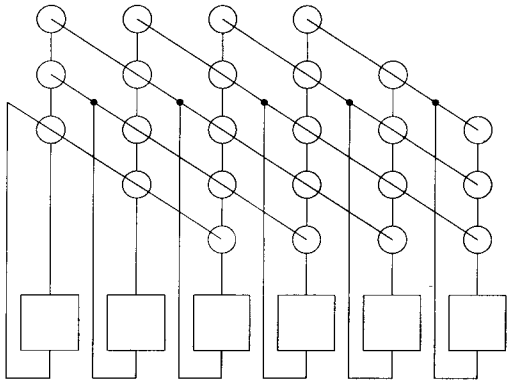


FIG. 9. — Advanced architecture for the one-dimensional neighborhood network.

Architecture évoluée pour réseau à voisinage unidimensionnel.

the *postsynaptic signals* with a summation line. If the synapse circuits are designed to deliver a current, these currents are summed in a node, the summation line.

The second advantage of analog circuit designs is the possibility to perform a multiplication with just a few transistors, e.g. with a transconductance multiplier. The multiplication of the input value with a so-called *weight* is the synaptic transmission function mostly used in today's static neural network models. One problem of the analog circuit design is the analog storage of this weight. This storage especially becomes difficult if a high accuracy is required. E.g., the *error-backpropagation* learning algorithm requires 12 to 16 bits accuracy of the weights [19].

For moderate accuracy requirements there are good ideas for analog circuit designs. At the University of Dortmund, e.g., an integration density of 64×64 connec-

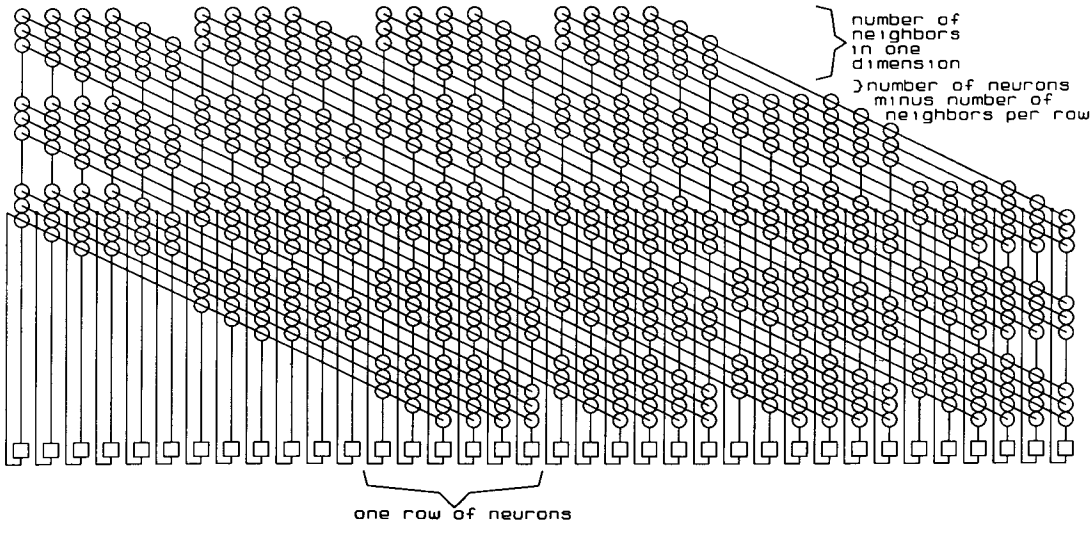


FIG. 10. — Advanced architecture for the two-dimensional neighbourhood network (example : 6 × 6 array of neurons, each connected to first and second neighbours).

Architecture évoluée pour réseau à voisinage bidimensionnel (par exemple un réseau de 6 × 6 neurones dont chacun est connecté au premier et au deuxième voisins).

tion elements of an associative learning matrix has been achieved in 2.5 μm-technology [20]. Such a learning matrix needs bivalent weights only, i.e., two processing elements are connected or not.

For trivalent weights — positive, negative or no connection — the problem is to match the positively and negatively weighted current. If p- and n-channel-transistors are used for the positive and negative current, respectively, the mismatch can be adjusted with the W/L-ratio. However, if there are many connections, the mismatch due to tolerances can lead to trouble. An idea to decrease the difference of the currents by using transistors of one type and two summation lines — one for positively and one for negatively weighted currents — is described in [22] (Fig. 11).

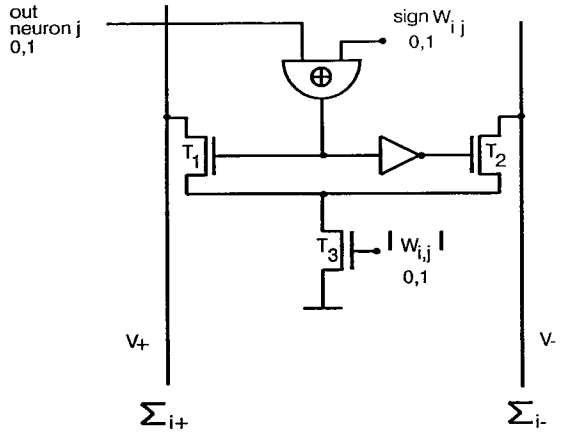


FIG. 11. — Analog synapse circuit for trivalent weights with transistors of one type for positive and negative weights [22].

Circuit synaptique analogique à pondération trivalente réalisé avec des transistors d'un seul type pour les poids positifs et négatifs.

For more accurate weights more accurate analog storage principles have to be taken into account, e.g. the storage of an analog voltage on a capacitor. In this case the reverse current over the drain-substrate junction discharging the storage capacitance has to be considered. On the one hand, there are concepts to reduce this discharging current by decreasing the voltage at the drain-substrate diode (Fig. 12) [16]. On the other hand, there is a concept to periodically refresh the stored voltages without the need to address each storage capacitor (Fig. 13) [8]. This is done with a triangular waveform and a clock signal connected to all storage

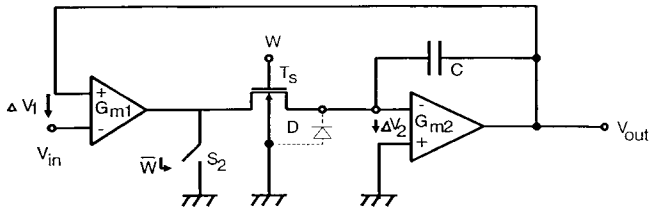
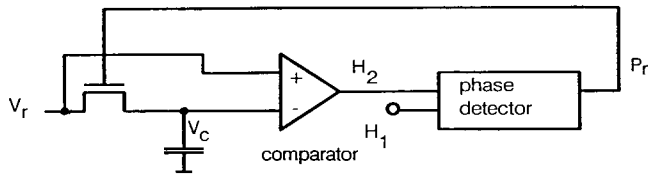


FIG. 12. — Analog storage cell with reduced current in reverse direction over the drain-substrate-diode [8, 16].

Cellule mémoire analogique à courant réduit en sens inverse à travers la diode drain-substrat.



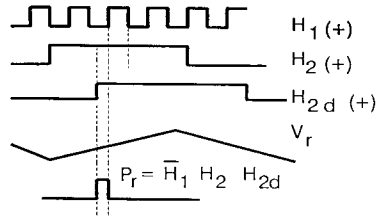


FIG. 13. — Concept for periodical refreshing of the stored charges [8].

Principe de rafraichissement périodique des charges stockées.

elements. The control circuit of each storage element regenerates the stored voltage if and only the triangular signal exceeds the stored voltage during a clock = high period.

Another possibility to store analog values even for longer times is the storage of charges on analog Eeprom-cells. Their capacitances are not nearly as discharged by any leakage currents so that charges can be stored for years. This kind of weight storage is also used for the Intel-chip *Etann* which is already commercially available. 64 neurons and 10.240 synapses are integrated on this chip. It occupies an area of 0.6 cm^2 in 1 μm-technology. A short-term accuracy of 8 bits and a long-term accuracy of 4 to 5 bits is mentioned in [9].

However, the exact writing of charges onto the floating Eeprom-gate is problematic. In [23] a circuit is proposed that approximates the desired value via a damped sigmoidal voltage. Further problems with analog Eeprom-storage-cells are the slow writing of values (order of micro- to milliseconds) and the limited number of write cycles. The latter makes long learning phases impossible.

A concept that makes use of digital and analog circuit design techniques is to perform the neural algorithms in pulse-coded arithmetics. The multiplication is done by an AND-operation of an input pulsestream with a weighting pulsestream, the pulsewidth of which represents the weight. The summation is approximated by an OR-function of the *postsynaptic* pulsestreams. The subsequent integration of the resulting pulsestream is an analog value for the *neuron activity*. A test chip contains 10 neurons and 100 synapses with 5 bits weight accuracy on an area of 174 × 73 mm^2 per synapse in 3 μm CMOS technology [14].

IV. POSSIBILITIES OF DIGITAL HARDWARE IMPLEMENTATIONS

In digital hardware concepts accuracy problems can be overcome by hardware effort. Then all known principles of digital circuit design can be applied to the storage of parameters and to the algorithms, too [1, 3, 5, 15, 17, 21]. Exemplarily, the concept for a general-purpose *neuroemulator* [2] will be shown here. For the learning phase and for the recognition phase, too, the most computing time intensive parts of the neural algoritms are matrix-matrix- and matrix-vector-multiplications. The latter can be transformed to matrix-matrix-multiplications, too, by combining a set of input vectors to an input matrix. In [2] a systolic processor for these matrix-matrix-multiplications is described. By a clever sorting of the input- and weight values this processor allows a partially serial, cascadable and fast computation of the neural algorithms.

V. A DIGITAL HARDWARE APPROACH TO A DYNAMIC NEURAL NETWORK

Measurements at the nervous system indicate that associative feature linking is coded via synchronization of temporal signals [6]. A dynamic neuron model, the Marburg model (Fig. 14), has been deduced from these measurements. With this model it is possible to imitate the synchronization effects. In contrast to the hardware approaches to conventional static neuron models shown above, the development of a digital emulator for this dynamic neural network will be explained in this chapter.

By exploiting the synchronization effects measured, neural networks consisting of biology-oriented dynamic neuron models, e.g. the Marburg model, offer new possibilities for associative pattern recognition [18]. The synapses of the Marburg model, i.e. the interconnections between the model neurons, are not just multiplicative weights but perform a dynamic transmission function described by two parameters, namely an amplitude and a delay parameter. The neural output function is a dynamic function, too, creating output pulses of constant duration and amplitude. Information is coded in the temporal arrangement of pulses.

Due to the internal feedbacks of the synapse function and the neural output function, the simulation of this dynamic neural network model is more computing time intensive than the simulation of static models. That is the reason specific hardware for the fast and flexible emulation of this model is urgently needed especially to develop and test learning strategies. The emulation results have to be reproducible and comparable to the simulations performed up to now. These requirements are fulfilled by our cascadable digital circuit.

In contrast to other biology-oriented neuron models, the Marburg model has two different types of inputs, namely feeding and linking inputs. The network architectures used with this model are multilayer networks with feedforward via feeding inputs and feedback via linking inputs. These architectures are inspired by hypercolumns found in the visual cortex. Due to these special architectural requirements, two types of synaptic matrices, one for feeding and one for linking synapses, have to be considered. The summation lines of both matrices would have to be connected to pins to achieve a flexible cascadability for both, feeding and linking synapses. Instead, a standard chip solution was chosen with only one matrix that can be configured to be feeding or linking. From this, one input of the feeding/linking-

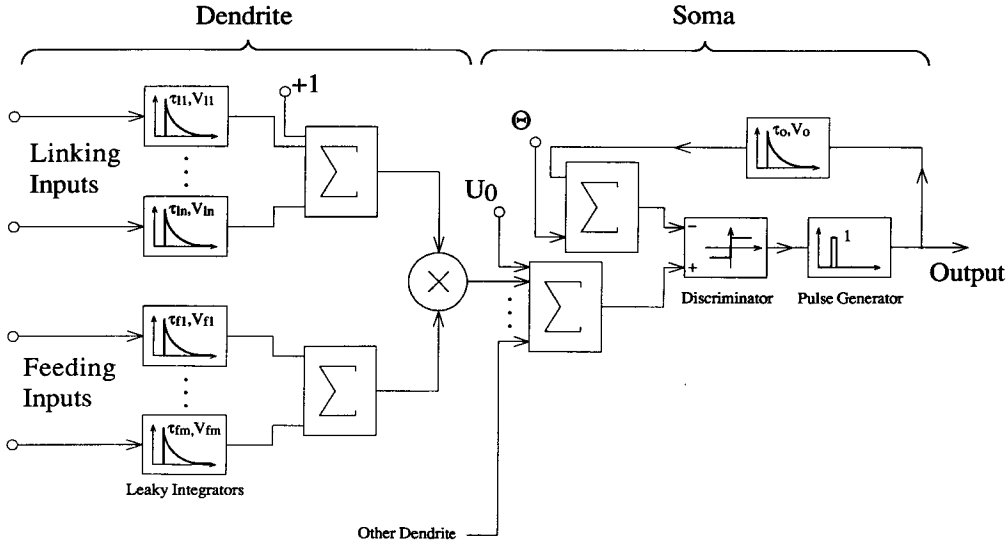


FIG. 14. — The Marburg model [16].

Le modèle de Marburg.

multiplier has to be connected to pins. The +1 for the sum of linking inputs (Fig. 14) can be fed to the cascadability input configuring synapses to be *linking*.

The synapse of the Marburg model can be emulated by a first order recursive digital filter. The main problems of the digital circuit design is the enormous area required for a single synapse and the number of pins required to transmit the sum of the postsynaptic signals over the chip borders for cascadability. Both problems can be solved by allowing a partially serial processing. If a defined number of synapses are processed serially and the processing is shifted in time, the number of multipliers and adders required for the synaptic transmission functions and for the neural output functions and the pin number required is divided by the number of synapses serially processed (Fig. 17). Furthermore, the partially serial processing results in a very dense and regular layout will be shown.

The starting point of the circuit design is the serial *16-synapses* building block. With respect to the bit accuracy requirements of the Applied Physics and Biophysics Department of the Marburg University, we decided to process 16 synapses serially (Fig. 15). Besides the multiplier, the most area consuming parts are the parameter memories, i.e. $2 \times 16 \times 8$ bits. Since the partially serial processing does not require a fast multiplication but a high throughput-rate and the storage of intermediate results, the multiplier is put into practice by a two-dimensionally pipelined 16×8 array multiplier. Pipelining offers the possibility to store the parameters in pipeline register loops, consisting of two minimal inverters and two transmission gates per stage. Parameters are written by connecting the loops of all synapse building blocks to one single loop and by putting in the parameter values serially. By this, any addressing overhead is avoided. The pipeline registers for the parameter storage for the feed back of the multiplication result are included in the layout of the multiplier cells. Furthermore, two wires for the summation of the postsynaptic signals are layed out inside the multiplier cell, the reason of which will be shown later. The adder and the AND-gate in Figure 15 perform a one-bit-multiplication, so that their circuit consists of multiplier cells, too. The multiplier cell described is the main building block for a regular abutment design of the whole synapse building block (Fig. 16).

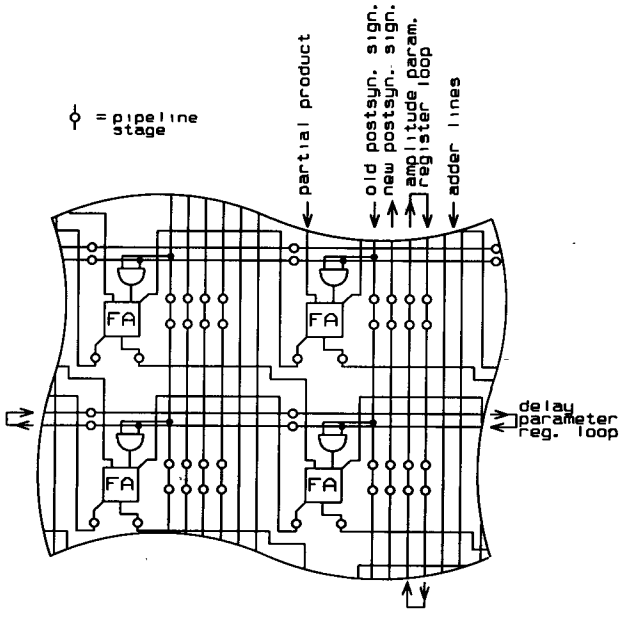


FIG. 16. — Regular circuit structure to achieve an abutment layout of the synapse building block.

Structure régulière de circuit pour disposer les éléments du bloc synaptique de base.

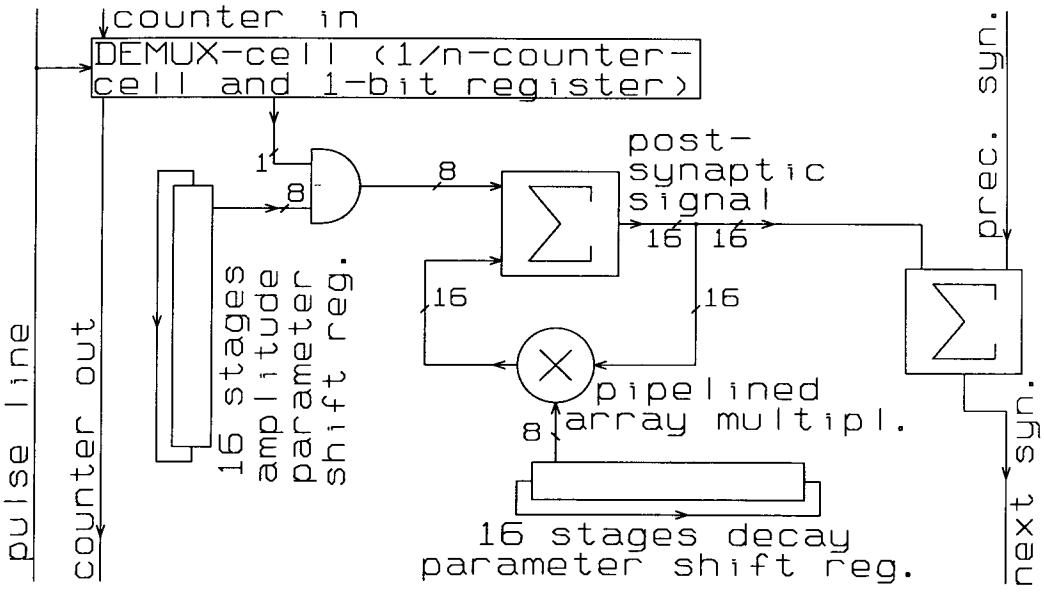


FIG. 15. — Functional schematic of the « 16-synapses » building block.

Schéma fonctionnel du bloc de base à 16 synapses.

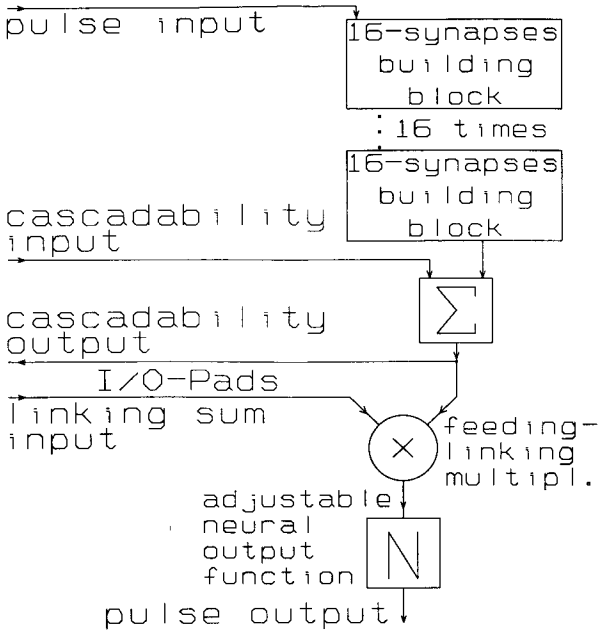


FIG. 17. — Block diagram of the emulator for biology-oriented neural networks.

Schéma de principe de l'émulateur de réseaux neuronaux modélisant des systèmes biologiques.

The 16-synapses building block described is regarded as a row of a 16 × 16 synapse matrix. The next step is to sum up the postsynaptic signals. Due to the serial processing, this summation can be done serially, too. The adder stage is layed out beneath the 16-synapses building block and connected to the preceding synapse by the wires, which have been included inside the multiplier cells as mentioned above. By this, the whole synapse matrix can be designed by an abutment of the 16-synapses building blocks. This abutment layout strategy can easily be adapted to larger matrices and higher bit accuracies by increasing the number k of synapses serially processed, i.e. increasing the number of multiplier cells inside the k-synapses building block.

A further special feature of the circuit is the possibility to adjust the shape of the output pulse — starting delay, pulsewidth, and a refractory time. This can be on one hand, used to adjust the temporal accuracy of the emulation, because the pulsewidth is a measure for the number of samples per time unit. On the other hand, the output pulse's delay, which is desired and biologically plausible, can be used to achieve a delayless cascadability. By decreasing the delay, the calculation start of cascaded chips can be shifted in time, so that there are no timing problems and no delays in network architectures containing feedback.

The chip is designed in 1.6 μm ES2 CMOS technology on an area of 88 mm^2 with 120 pads. It virtually contains an emulator for 16 Marburg model neurons with 16 synapses per neuron including all parameter memories. Several of these chips can be cascaded to fully interconnected networks and to multilayer networks (with or without feedback) without any delay overhead. Due to the nature of the neuron model, the power of the neural network circuit presented cannot be measured by interconnections per second. It performs 1 sample per 16 clock cycles not depending on the number of cascaded chips, on the number of neurons, nor on the number of interconnections.

VI. SUMMARY

Different chip architectures and circuit designs for the implementation of neural networks on silicon have been shown. It should have become clear, that especially in the analog area restrictions according to accuracy have to be considered. Digital circuits can fulfill the accuracy requirements by hardware effort. To find a compromise between speed and hardware effort partially serial concepts are taken into consideration. However, with today's digital processing speeds this does not have to be a restriction.

Manuscrit reçu le 3 février 1993.

REFERENCES

[1] ALLA (P. Y.), OUALI (J.), SAUCIER (G.), MASSE-NAVETTE (L.), TRILHE (J.). Silicon compilation of neuro-ASICs supported by distributed and synchronous neural network architecture. *Proc. of the 2nd Intern. Conf. on Microelectronics for Neural Networks* (1991), pp. 341-346.

[2] BEICHTER (J.), BRÜLS (N.), MEISTER (E.), RAMACHER (U.), KLAR (H.). Design of a general-purpose neural signal processor. *Proc. of the 2nd Intern. Conf. on Microelectronics for Neural Networks* (1991), pp. 311-315.

[3] BLAYO (P.), HURAT (P.). A VLSI systolic array dedicated to hopfield neural network. *In :* Delgado-Frias (J. G.), Moore (W. R.) (eds.) : VLSI for artificial intelligence. *Kluwer Acad. Publ.* (1988), pp. 255-264.

[4] CHUA (I. O.), YANG (L.). Cellular neural networks : theory. *IEEE Transactions on Circuits and Systems* (Oct. 1988), **35,** n° 10, pp. 1273-1290.

[5] DISTANTE (F.), SAMI (M. G.), STEFANELLI (R.), STORTI-GAJANI (G.). A proposal for neural macrocell array. *In :* Sami (M.), Calzadilla-Daguerre (eds.); Silicon architectures for neural nets. *Elsevier Science Publ.* (1991).

[6] ECKHORN (R.), REITBOECK (H. J.), ARNDT (M.), DICKE (P.). A neural network for feature linking via synchronous activity : results from cat visual cortex and from simulations. *In :* Cotterill (R. M. J.) (ed.); Models of brain function. *Cambridge University Press* (1989).

[7] HARRER (H.), NOSSEK (J. A.), SEILER (G.), STELZL (R.). An analog CMOS compatible convolution circuit for analog neural networks. *Proc. of the 2nd Intern. Conf. on Microelectronics for Neural Networks* (1991), pp. 231-241.

[8] HOCHET (B.), PEIRIS (V.), ABDO (S.), DECLERCQ (M. J.). Implementation of a learning Kohonen neuron based on a new multilevel storage technique. *IEEE Journal of Solid-State Circuits* (Mars 1991), **26,** n° 3, pp. 262-267.

[9] HOLLER (M.), TAM (S.), CASTRO (H.), BENSON (R.). An electrically trainable artificial neural network (Etann) with 10 240 « Floating gate » synapses. *Proc. of the Intern. Joint Conf. on Neural Networks*, Washington D.C. (juin 1989), **2,** pp. 191-196.

[10] JOHANNESMA (P. I. M.), VAN DEN BOOGAARD (H. F. P.). Stochastic formulation of neural interaction. *Acta Applicandae Mathematicae* (1985), **4,** pp. 201-224.

[11] KLAR (H.), RAMACHER (U.). Microelectronics for artificial neural networks. *VDI-Verlag*, Düsseldorf (1989).
[12] KOHONEN (T.). Self-organization and associative memory. *Springer Verlag*, Heidelberg (1984).
[13] LIPPMANN (R. P.). An introduction to computing with neural nets. *IEEE ASSP Magazine* (Apr. 1987), pp. 4-22.
[14] MURRAY (A. F.). Pulse arithmetic in VLSI neural networks. *IEEE Micro* (Dec. 1989), pp. 64-74.
[15] MYERS (D. J.), VINCENT (J. M.), ORREY (D. A.). Hannibal : a VLSI building block for neural networks with on-chip backpropagation learning. *Proc. of the 2nd Intern. Conf on Microelectronics for Neural Networks* (1991), pp.171-181.
[16] NAHEBI (M.), WOOLEY (B.). A 10-bit video BiCMOS track-and-hold amplifier. *IEEE Journal of Solid-State Circuits* (Dec. 1989), **24,** n° 6, pp. 1507-1516.
[17] PERSONNAZ (L.), JOHANNET (A.), DREYFUS (G.), WEINFELD (M.). Towards a neural network chip : a performance assessment and a simple example. *In :* Personnaz (L.), Dreyfus (G.) (eds.); Neural networks from models to applications. *ESPCI*, Paris (1988).
[18] REITBOECK (H. J.), ECKHORN (R.), ARNDT (M.), DICKE (P.). A model of feature linking via correlated neural activity. *In :* Haken (H.) (ed.); Synergetics of cognition. *Springer Verlag*, Berlin (1989).
[19] REYNERI (L. M.), FILIPPI (E.). An analysis on the performance of silicon implementations of backpropagation algorithms for artificial neural networks. *IEEE Trans. on Computers* (Dec. 1991), **40,** n° 12, pp. 1380-1389.
[20] RÜCKERT (U.). An associative memory with neural architecture and its VLSI implementation. *Proc. of the 24th Annual Hawaii Intern. Conf. on System Sciences* (1991), **1,** pp. 212-218.
[21] SCHWARZ (M.), HOSTICKA (B.), RICHERT (P.), KESPER (M.). Concept of neural hardware for parallel image processing. *Proc. of the 1st Intern. Workshop on Microelectronics for Neural Networks* (1990), pp. 153-158.
[22] VERLEYSEN (M.), SIRLETTI (B.), VANDEMEULEBROECKE (A. M.), JESPERS (P. G. A.). Neural networks for high-storage content-adressable memory : VLSI circuit and learning algorithm. *IEEE Journal of Solid-State Circuits* (juin 1989), **24,** n° 3, pp. 562-569.
[23] VITTOZ (E.), OGUEY (H.), MAHER (M. A.), NYS (E.), DIJKSTRA (E.), CHEVROULET (M.). Analog storage of adjustable synaptic weights. *In :* Ramacher (U.), Rückert (U.) (eds.); VLSI design of neural networks. *Kluwer Acad. Publ.* (1991).