

Digital Neurohardware: Principles and Perspectives

T. Schoenauer, A. Jahnke, U. Roth and H. Klar
Institute of Microelectronics, Technical University of Berlin
Jebensstr.1, Sekr. J13, D-10623 Berlin
Phone: +49 30 314-22640, Fax: +49 30 314-23029
E-mail: tim@mikro.ee.tu-berlin.de

Abstract

Over the past years a huge diversity of hardware for artificial neural networks (ANN) has been designed. This paper gives an overview of the different types of digital ANN hardware and discusses a few example architectures. The development of digital neurohardware is driven by the desire to speed-up the simulation of ANN and/or to achieve a better performance-to-cost ratio than general-purpose systems. The most basic approach to speed-up ANN algorithms is to parallelize processing. Therefore, strategies to efficiently map neural algorithms to parallel hardware are outlined. On the other hand, the performance of general-purpose sequential hardware platforms, such as workstations and PCs, has also dramatically improved in the last couple of years. Hence, at the end we raise the question, if there is still a need for neurohardware and discuss future perspectives.

Keywords: Neurocomputer, Neuro-Accelerators, Neurochips, Parallel Processing

1 Introduction

General-purpose computers are traditionally based on the von-Neumann architecture which is sequential in nature. Artificial neural networks on the other hand profit from massively parallel processing. To provide hardware platforms for efficient simulation of ANN, a tremendous variety of hardware has been designed. One basic concept of digital neurohardware designs is to connect many simple processors together to allow for fast parallel processing. The implementation beyond this common idea however differs strongly. In order to get an overview of existing digital neurohardware systems, a taxonomy is introduced in section 2. Parallel hardware is most profitable if none of the processing elements are idle. To approach such an optimal load balance and to minimize communication of processors, the mapping schemes of neural networks on parallel computer are described in section 3. In section 4, three examples of neurocomputers are given: the well-known CNAPS, SYNAPSE and NESPINN are presented. Advantages as well as limitations of all three systems are pointed out. Finally, the question is raised if there is still a need for digital neurohardware and future perspectives are discussed.

2 Classification of Digital Hardware for Artificial Neural Networks

Today one can choose from a wide range of neural network hardware. Designs differ in terms of architectural approaches, such as neurochips, accelerator boards and multi-board neurocomputers, as well as concerning the purpose of the system, such as the ANN algorithm(s) and the system's versatility. A few surveys of neural network hardware have been published [2],[10],[12]. Due to the diversity of neurohardware, the choice of the optimal hardware platform to implement an ANN algorithm is difficult. Hence, some kind of taxonomy of neurohardware designs would allow a better evaluation of existing designs. However the various classifications proposed in literature [1], [2], [11], [3] differ and show that such categorization is subjective. In the context of this paper, we favor the taxonomy proposed in [1]. Digital neurohardware can be classified by the: system architecture, degree of parallelism, typical neural network partition per processor, inter-processor communication network and numerical representation. The taxonomy is illustrated in Fig. 1.

Type of System: System architectures range from single stand-alone neurochips to full-fledged neurocomputers. Single neurochips may serve as microprocessor peripherals (e.g.[13]) or can be used on-board as a stand-alone chip (e.g. [14]). Accelerator-boards are connected to a PC or a workstation and generally provide more flexibility than a neurochip. Accelerator boards are either equipped with general-purpose processors (e.g. [15]) or with custom processors. Neurocomputer are even more powerful systems consisting of either neurochips and/or commercial processors. Adaptive Solutions' CNAPS [4] as well as Siemens' SYNAPSE [8] are examples for neurocomputers using dedicated neurochips. However, there are also accelerator boards available using the same

neurochips.

Degree of Parallelism: The number of processing elements yield the degree of parallelism of a system. The more parallel units there are, the faster data is processed. However, parallelism is expensive in terms of chip area or chip count. Therefore highly parallel systems usually employ simpler processing elements. The parallelism can be rated from only a few processing elements referred to as coarse-grained up to almost a one-to-one implementation of neural processing nodes called massive. There are no definite borders between these different categories as

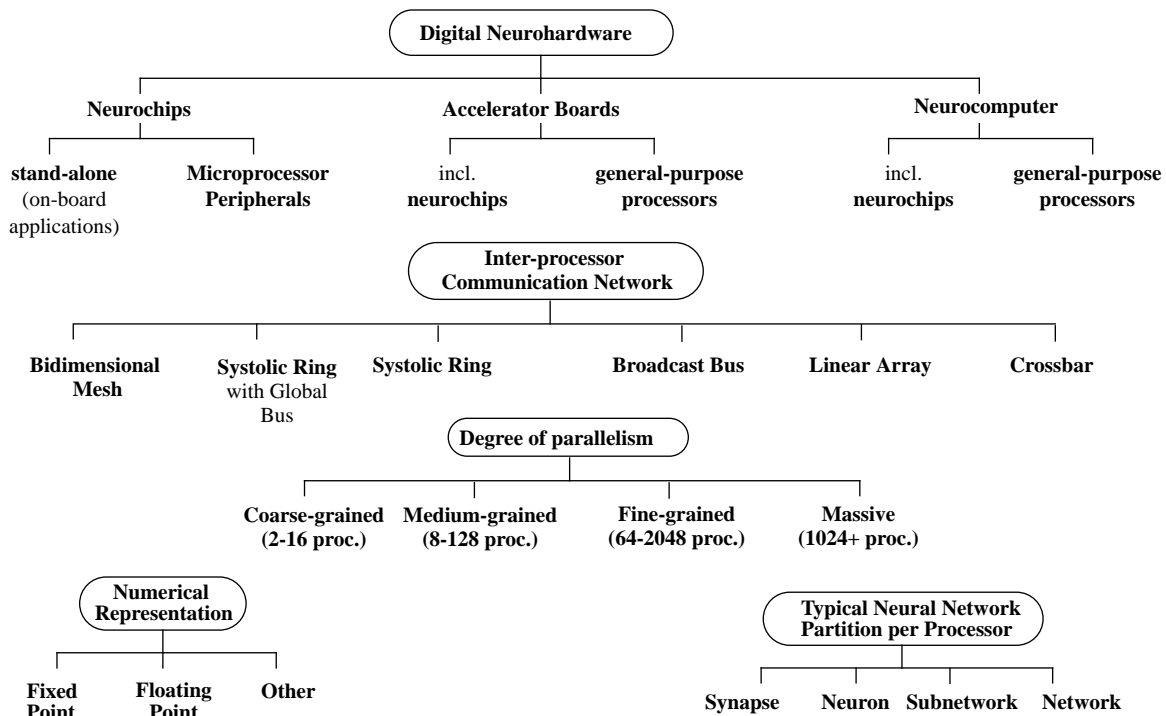


Figure 1: Taxonomy of digital neurohardware (after [1])

illustrated by the overlapping intervals of the adjacent classes.

Inter-processor Communication Network: Parallel processing elements only speed-up the computation when they do not run idle. Thus, for the system performance it is crucial that the inter-processor communication network provides the processing elements with sufficient data. Broadcast bus, linear array, systolic ring, crossbar and bidimensional mesh are the most frequently encountered communication networks of ANN systems.

Numerical Representation: In general, neural networks have low-precision requirements, even though the exact specification is algorithm and application dependent. Digital neurohardware can profit from this property by using fixed-point arithmetic with reduced precision. Fixed-point implementations are less complex and less area consuming than floating-point arithmetic and therefore their use helps to reduce system cost. Other numerical representations such as stochastic pulse stream coding are also used (e.g. [14]).

Typical Neural Network Partition per Processor: Depending on the hardware architecture and the neural algorithm, the neural network needs to be mapped to the parallel processing elements. Either this is done by n(euron)-parallel mapping, s(ynapse)-parallel mapping or by mapping a pattern of subnetwork or even the total network to each PE. In the following section we discuss the mapping schemes in more detail.

3 Mapping Neural Networks on Parallel Computers

How can we map a specific neural network on a parallel computer to achieve the maximum performance? The key concepts of an efficient mapping are load balancing, minimizing inter-PE communication and minimizing synchronization between the PEs. Furthermore, the mapping should be scalable both for different network sizes, and for different number of processing elements. In Fig. 2 the weight matrix presentation of a simple neural network (four neurons with four synapses each) is shown in the middle, while the left side shows the conventional presentation of the same network. The rectangle N in the mid part of Fig. 2 denotes the activation function of the

neuron. The circle w_{ij} represents the computation of the synapse: $y_i = w_{ij} u_j + y_{i-1}$ where y_{i-1} is the result from the preceding synapse.

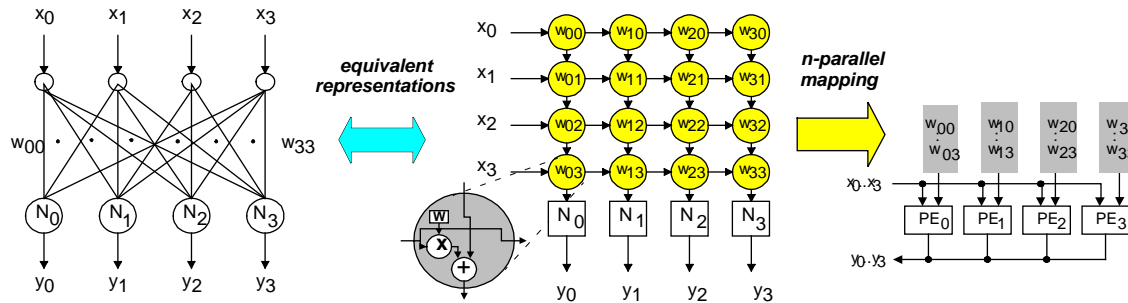


Figure 2: Presentation of a neural network: conventional (left), weight matrix (middle), mapped N-parallel (right)

Let us assume we would like to simulate a network with N neurons on a parallel computer with N_{PE} PEs. We will illustrate the mapping with $N = 4$ and $N_{PE} = 4$ in the following.

Neuron-parallelism: The synapses of one neuron and the output function are mapped to the same PE and N_{PE} neurons are computed in parallel (Fig. 2). This mapping scheme is called *neuron-parallel* or simply *n-parallel*.

Synapse-parallelism: Instead of mapping the synapses of one neuron to the same PE they can be mapped to different PEs. So, N_{PE} synapses -not necessarily from the same neuron- will be computed in parallel. The output of a neuron can be processed either on a separate PE serially (mid part of Fig. 3) or can be distributed over the PEs (right part of Fig. 3). We will call the first kind of mapping s(ynapse)-parallel and the latter sn-parallel.

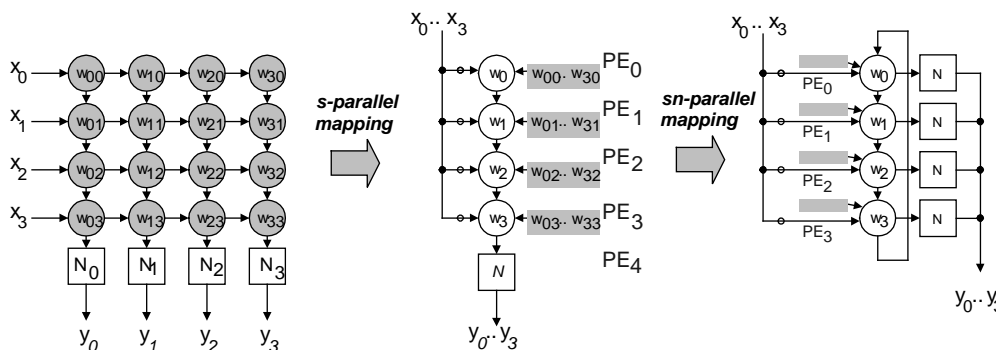


Figure 3: S- and SN-Parallelism

Pattern-parallelism: In order to reduce the required communication bandwidth p(attern)-parallelism has been used for conventional networks such as MLPs with backpropagation learning. Each PE of the parallel processor simulates the output of the network for a different input pattern.

4 Examples of Digital Neurohardware

4.1 CNAPS

One of the most well known commercially available neurocomputers is the CNAPS (Connected Network of Adaptive Processors) from Adaptive Solutions. The basic building block of the CNAPS system is the neurochip N6400 [4]. As shown in Fig. 4, the N6400 itself consists of 64 processing elements (referred to as processing nodes PN) which are connected by a broadcast bus in a SIMD (Single Instruction Multiple Data) mode. Two 8bit buses allow the broadcasting of input and output data to all PNs.

One of the big advantages of the CNAPS architecture is the scalability of the system: due to the broadcast bus, inter-processor communication and the SIMD mode, additional N6400 chips can be easily added as depicted in Fig. 4. The standard CNAPS system consists of a common sequencer chip and four processor chips (systems with up to eight chips and altogether 1064 PNs are available). The regularity of the broadcast bus structure is exploited also in another way in the CNAPS system: the N6400 die measures about one square inch with more than 13 million transistors integrated. The yield is kept at an acceptable level by introducing redundancies and reconfiguring faulty elements after fabrication [5]. Out of 80 PNs integrated 64 PNs are used after test and reconfiguration, resulting in a 90% yield.

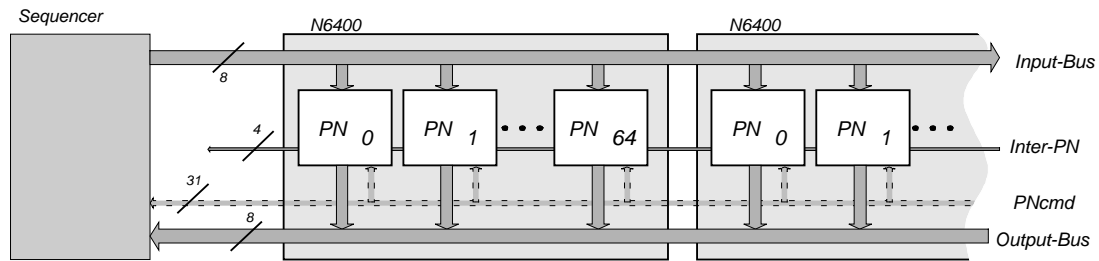


Figure 4: SIMD-Architecture of the CNAPS

The PNs are designed like simple DSPs including fixed-point adder and multiplier. Each PN is equipped with 4KByte local on-chip SRAM which needs to hold the weights. The size of the local memory is the bottle-neck for large networks: once the connectivity cannot be stored locally anymore a communication via the broadcast bus becomes necessary. Of course the system performance drops dramatically when 64 PNs try to communicate over two 8bit buses. Since the two data buses do not allow an efficient communication between PNs, networks must be mapped n-parallel onto the CNAPS. When employing backpropagation learning each processing node has to store not only the weight matrix but also the inverse as well. In that case the size of networks the CNAPS architecture can handle is smaller.

However, the versatile character of the PN provides programmability for a broad range of algorithms: the possibility of implementing several algorithms including backpropagation [6] and Kohonen self-organizing feature maps [7] as well as image processing algorithms. Also, for convenient programming CNAPS tools include a C-compiler with extensions to take full advantage of the parallel architecture. According to the previously mentioned taxonomy of ANN hardware, one would consider the complete CNAPS system a neurocomputer built of neurochips. However, the N6400 has also been used to build accelerator boards.

4.2 SYNAPSE-1

Siemens' MA-16 neurochip is the basic building block for the neurocomputer SYNAPSE-1 (Synthesis of Neural Algorithms on a Parallel Systolic Engine). MA-16 is designed for fast 4×4 matrix operations with 16bit fixed point precision [8]. Multiple MA-16 chips can be cascaded to form systolic arrays. This way inputs and outputs are passed from one MA-16 chip to another in a pipelined manner ensuring an optimal throughput.

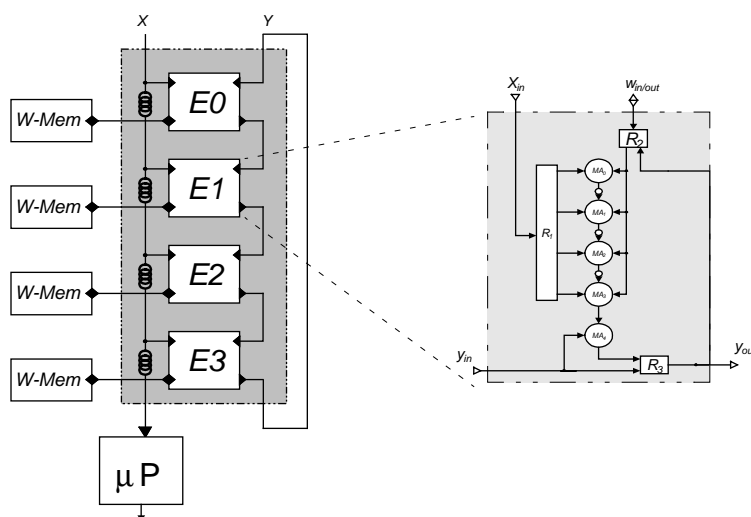


Figure 5: Scalar product chain of the MA-16 chip

The SYNAPSE-1 consists of eight MA-16 chips connected in two parallel rings controlled by two Motorola MC68040 processors. Weights are stored in an off-chip DRAM which amounts to 128MByte and can be further expanded up to 512MByte. The neural network is mapped sp-parallel for the forward phase and np-parallel for the learning phase. The neuron transfer functions are calculated off-chip using look-up tables. Especially the high capacity of the on-line weight memory qualifies the SYNAPSE-1 for complex applications. Like the CNAPS,

SYNAPSE-1 is not dedicated to specific algorithms. Several networks have been mapped onto SYNAPSE-1, e.g. backpropagation and Hopfield networks. In opposite to the simple SIMD architecture of the CNAPS, programming the SYNAPSE-1 is difficult. The fairly complex processing elements and the 2-dimensional structure of the systolic array hinder a straight-forward programming even though a neural Algorithmic Programming Language is available.

4.3 NESPINN

The previous two neurocomputers discussed, the CNAPS and the SYNAPSE were designed for a wide range of neural network algorithms. In the following example, the NESPINN (Neurocomputer for Spiking Neural Networks) designed at the Institute of Microelectronics of the Technical University of Berlin is presented. This architecture is optimized more strictly to a certain class of neural networks: spiking neural networks. Spiking neural networks model neurons on a level relating more closely to biology. They do not only incorporate synaptic weighting, postsynaptic summation, static threshold and saturation, but also computation of membrane potentials, synaptic time delays and dynamical thresholds.

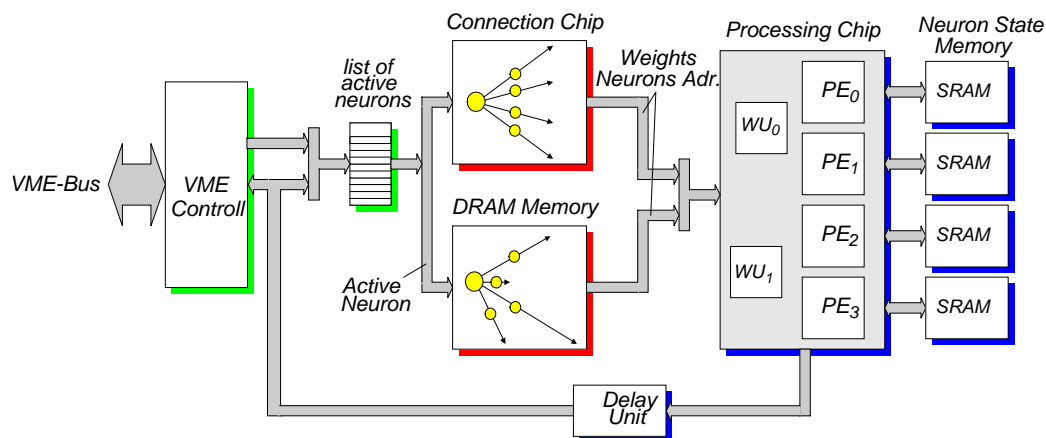


Figure 6: Architecture of a NESPINN-Board

One NESPINN-Board is designed to compute about 10^5 programmable neurons in real-time [9]. A NESPINN-board contains two ASIC (Application Specific Integrated Circuit): one connection chip and one processing chip as shown in Fig. 6. The processing chip includes 4 Processing Elements (PE) with 2KByte on-chip SRAM each and at least 64KByte external memory each. NESPINN operates in a Mixed SIMD/dataflow mode and the network is mapped n-parallel. The ability to compute up to 10^5 "biological" neurons is achieved by taking advantage of characteristics of spiking neural networks: low network activity and a sparse connectivity. For example, membrane potentials are stored in the external Neuron State Memory (see Fig. 6). Due to a low network activity a considerable percentage of the membrane potentials is negligible. Thus, data from the Neuron State Memory is only read or written if it exceeded a certain threshold ensuring that all processed potentials are of relevance.

Even though for the simulation of spiking neurons, this design promises a performance only rivaled by supercomputers, the system's efficiency is basically limited to this class of neuron models. Although the NESPINN-System is capable of simulating simpler models like the Multilayer Perceptron (MLP), the asset in performance compared to conventional hardware vanishes quickly.

5 Neurohardware: Problems & Perspectives

Among the challenges digital neurohardware faces today the competition with general-purpose hardware is probably the toughest one: computer architecture is a highly competitive domain which advances at an incredible pace. The area of ANN hardware on the other hand is not yet as commercialized as general-purpose hardware. Also digital neurohardware tends to be more algorithm specific. This requires a good knowledge about algorithms as well as system design and leads to a high time-to-market. Therefore, general-purpose computers can profit more often from advances in technology and architectural revisions. Also, in many other respects general-purpose hardware seems to be more user-friendly: it is not bound to algorithmic a-priori-assumptions and therefore offers high flexibility. Uniform programming interfaces exist for general-purpose hardware. This can be important not only to get a better start when programming a system, but also to allow reusability when moving on to the next hardware generation.

On the other hand, there are ANN problems, exceeding the computational capabilities of workstations or PCs such as real-time applications, the simulation of large networks or networks employing very complex neuron models. For these applications neurohardware is attractive, since many aspects of the userfriendliness vanish for supercomputers. Besides the difficult programming of supercomputers, they are very expensive. Neurohardware might also provide a much better cost-to-performance ratio in such a case. Most vitalizing for the domain of ANN hardware however would be a paradigm successful enough in real-world applications to demand dedicated neurohardware for its potential advantages of a higher system integration, a better cost-to-performance ratio, a lower power consumption and a smaller size.

6 Conclusion

This paper gave an overview of digital hardware for neural networks. A classification of existing hardware helped to identify different approaches in neurohardware system design with their advantages and disadvantages. To make efficient use of the parallel hardware, the network must be mapped properly onto the platform. Therefore, we examined basic strategies to map neural networks on parallel computers. Three examples of digital neurohardware, the CNAPS, SYNAPSE and the NESPINN-system were presented. CNAPS is a very versatile and powerful platform for ANN, as long as the on-chip memories of the processing nodes can store the connectivity. SYNAPSE is an even more powerful neurocomputer consisting of vector processors in a systolic array. System complexity, however, restricts userfriendly programming. More than the other two neurocomputers, the NESPINN-design exploits more aggressively the characteristics of a certain class of neural networks. In any way, designing neurohardware is a complex and time consuming task which is also constantly challenged by improvements of a highly competitive traditional computer industry. Clearly an algorithmic success in the field of ANN would revive the area of neurohardware. Paradoxically, the capabilities of many paradigms are hardly known, yet these capabilities can only be tested when special hardware is available. As long as conventional hardware is not sufficient in such a case, there is still a need for neurohardware.

7 References

- [1] Jenne, P., 1997, *Digital Connectionist Hardware: Current Problems and Future Challenges*. In J. Mira, R. Moreno-Díaz, and J. Cabestany, editors, *Biological and Artificial Computation: From Neuroscience to Technology*, Volume 1240 of *Lecture Notes in Computer Science*. Pages 688-713. Springer, Berlin, 1997.
- [2] Heemskerk, J. N. H., 1995, *Overview of Neural Hardware*. In: *Neurocomputers for Brain-Style Processing. Design, Implementation and Application*, PhD Thesis, Unit of Experimental and Theoretical Psychology Leiden University, The Netherlands. Draft of this chapter is available via ftp: <ftp.mrc-apu.cam.ac.uk/pub/nn/murre/neurhard.ps>.
- [3] Ramacher, U., Rückert, U., 1991, *VLSI Design of Neural Networks*. *Kluwer Academic Pub.*, 1991.
- [4] Hammerstrom, D., 1991, *A highly parallel digital architecture for neural network emulation*. In: Delgado-Frias, J. G. and Moore, W. R. (eds.), *VLSI for Artificial Intelligence and Neural Networks*, chapter 5.1, pages 357-366. Plenum Press, New York, 1991.
- [5] Griffin, M., Tahara, G., Knorpp, K. et al., 1991, An 11 million transistor neural network execution engine. *IEEE Internation Conference on Solid-State Circuits*, pages 180-181, 1991.
- [6] Hammerstrom, D. and Nguyen, N., 1991, An Implementation of Kohonen's Self-Organizing Map on the Adaptive Solutions Neurocomputer. In: Kohonen, T. et al. (eds.), *Artificial Neural Networks, Proceedings of the ICANN'91*, Amsterdam: Elsevier, pages 715-720, 1991.
- [7] McCartor, H., 1991, Backpropagation Implementation on the Adaptive Solutions CNAPS Neurocomputer chip, *Proceedings of NIPS-3, Advances in Neural Information Processing Systems 3*, Lippmann, R. et al. (eds.), pages 1028-1031, Morgan Kaufmann Pub., 1991.
- [8] Ramacher, U., 1992, SYNAPSE - A neurocomputer that synthesizes neural algorithms on a parallel systolic engine. *Journal of Parallel and Distributed Computing*, 14(3): pages 306-318, Mar. 1992.
- [9] Jahnke, A., Roth, U. and Klar, H., 1996, A SIMD/Dataflow Architecture for a Neurocomputer for Spike-Processing Neural Networks (NESPINN). *MicroNeuro'96*, pages 232-237, 1996.
- [10] Glesner, M. and Pöschmüller, W., 1994, *An Overview of Neural Networks in VLSI*. Chapman & Hall, London, 1994.
- [11] Lindsey, C. and Lindblad, T., 1994, Review of Hardware Neural Networks: A User's Perspective. *Proceedings of 3rd Workshop on Neural Networks: From Biology to High Energy Physics*, Isola d'Elba, Italy, Sept.26-30, 1994. Also available from <http://www1.cern.ch/NeuralNets/nnwInHepHard.html>
- [12] Misra, M., 1997, Parallel Environments for implementing Neural Networks, *Neural Computing Survey*, Vol.1, 48-60, 1997. Also available from <http://www.icsi.berkeley.edu/~jagota/NCS>
- [13] Mauduit, N., Duranton, M. and Gobert, J. et al., 1992, Lneuro 1.0: Apiece of hardware LEGO for Building Neural Network Systems. *IEEE Transactions on Neural Networks*, 3, 3: 414-422, May 1992.
- [14] Oteki, S. et al., 1993, A digital neural network VLSI with on-chip learning using stochastic pulse encoding. *Proceedins of the International Joint Conference on Neural Networks*, 3: 3039-3045, Nagoya, Japapn, Oct. 1993.
- [15] HNC, 1993, High-Performance Parallel Computing. *SIMD Numerical Array Processor*, Data Sheet, San Diego, USA.