A Platform for Parallel Operation of VLSI Neural Networks

J. Fieres*, A. Grübl*, S. Philipp*, K. Meier, J. Schemmel, F. Schürmann
University of Heidelberg, Kirchhoff Institute for Physics,
Im Neuenheimer Feld 227, 69120 Heidelberg, Germany
E-mail: sphilipp@kip.uni-heidelberg.de
http://www.kip.uni-heidelberg.de/vision
phone: +49-6221-549897
*These authors contributed equally to this work.

***Abstract*— This paper presents a platform for the parallel operation of VLSI neural networks allowing to seamlessly map neural network topologies on distributed resources. The scalable approach provides fast isochronous communication channels transporting the neuron signals between single network modules. The network modules are printed circuit boards hosting a programmable logic with an embedded microprocessor core, memory, and a VLSI neural network ASIC. Currently, the modules are equipped with a mixed-signal neural network ASIC. For its McCulloch-Pitts type neurons a biologically inspired higher-level model of perception is adopted and demonstrated.**

I. Introduction

In recent years researchers in the field of artificial neural networks (ANN) have intensified their efforts in reconsidering the biological origin of neural networks. As a consequence more biologically plausible neuron models and learning strategies are investigated. Independently, higher-level models of perception were adopted with rather abstract network models [1] and recently with physiologically inspired models [2].

Within the SenseMaker [3] consortium, a European project, neuroscientists, psychologists, engineers and physicists work in collaboration in order to gain insight into the principles of information processing in neural systems. The goal is to design and implement an electronic system capable of merging sensory input of different modalities to generate a perceptual representation of the outside world. Through the course of this three year project several stages of hardware neural systems are developed. The mixed-mode neural network ASIC (application specific integrated circuit) HAGEN is used as the basis for the investigation of higher-level concepts and has been published earlier [4]. In this paper individual network modules are presented which consist of programmable logic with an embedded microprocessor core and a neural network ASIC. A dedicated backplane hosts 16 of these modules and allows high-speed communication between them. Due to the underlying network model and the provided signaling protocol a closed formula for the network response can be given. This allows not only to operate the modules in parallel but also to use the distributed neurons together in a larger network topology. A backplane fully equipped with network modules using the HAGEN ASICs provides about 4k McCulloch-Pitts neurons and 512k synapses. Eventually, this platform is ready to host a spiking neural network ASIC which will also allow to examine biological low-level principles.

The paper is organized in three sections. First, the parallel hardware system is presented. The following section describes how the concept of a system of distributed neurons can be used to form a consistent network model. In the last section it is demonstrated how a higher-level model of perception, the Neocognitron idea of Fukushima [1], can be implemented on networks of the HAGEN ANN type.

II. Hardware Implementation

A. Design Considerations

The concept for a platform operating VLSI (very large-scale integration) neural networks in parallel is based on the research done with the ANN ASIC HAGEN in single PC based systems. The need for larger scales can be met either by building a larger ANN ASIC or by using several of the existing ones in combination. The latter has been chosen since the potentials gained by a general parallel system extend the capabilities of the current ANN chip as well as maintaining the flexibility to be used with future ANNs or ASICs of a totally different kind.

The central components of the parallel system are highly integrated autonomous network modules [5], each containing all necessary components to interface one neural network chip: programmable logic, a local CPU, memory, and the ANN ASIC. In order to use the network modules in parallel, strategies are necessary to coordinate the distributed resources:

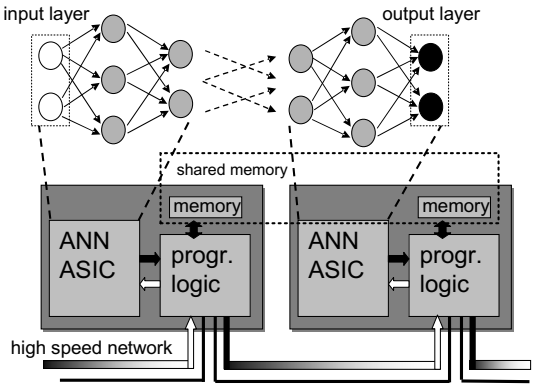


Fig. 1. Example of a large neural network, implemented using two ANN ASICs.

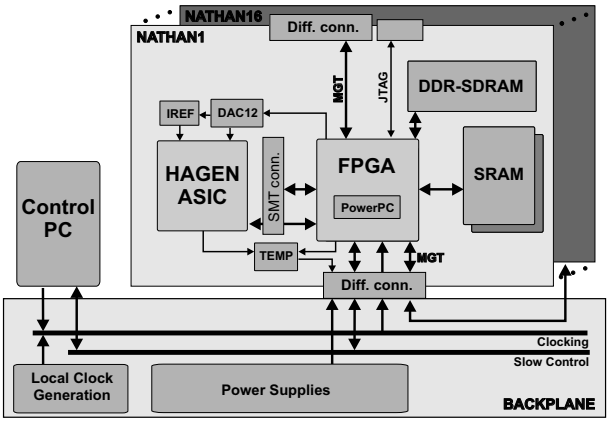


Fig. 2. Overall schematic of the hardware platform.

When operating on data streams, neural networks constantly require inputs and generate outputs. Since the used neural network chips accept digital inputs and produce digital outputs, the data can easily be transported by digital communication technologies. Thus, the outputs can be fed to another neural network chip, which for example implements an inner layer of a larger network. Maintaining this constant signal feed requires a carefully designed connectivity, especially, if the system shall remain scalable.

The necessary high connectivity between the individual network modules is ensured by a high speed transport network which interconnects the modules via a backplane. This network delivers neuron outputs from one chip to the input neurons of another chip. The global neural network will then consist of *virtual connections* between the chips using the high speed transport network (see Fig. 1). For a deterministic operation of the neural network it is important to ensure that the underlying transport network can deliver the data for every connection with a guaranteed latency. In clocked neural networks, the transport network has to deliver the data in successive neural network cycles.

Furthermore, it may be useful for the entities controlling the neural network operation (i.e., the controlling software and/or programmable logic) to exchange larger amounts of data. To have a convenient way doing so, the transport network is used to create a large virtual shared memory structure [6]. This allows to access memory on any module by its virtual memory address.

B. The HAGEN Chip

The ANN chip HAGEN contains 32768 analog synapses, distributed across four equally sized blocks. The synapses of each block are arranged in an array structure, connecting 128 binary inputs with 64 binary outputs, thus making up a fully connected single-layer feed-forward Perceptron consisting of McCulloch-Pitts neurons. The analog computations are fully confined within the blocks. Inputs and outputs are interfaced digitally, and the synaptic weights are converted by digital to analog converters. Thus, all communication is digital, ensuring data integrity while still exploiting the advantage of fast and highly integratable analog computing units. The network operates at constant frequency, where the outputs of all neurons are calculated once each network cycle. To set up recurrent networks, some of the outputs can be fed back to the input neurons of the same or of a different block, using on-chip configurable feedback lines. In clocked networks, a feedback connection introduces a time delay of multiples of a network cycle. All hardware feedbacks within a HAGEN chip have a delay of one cycle. However, the chosen network update scheme does not imply restrictions on the number of network cycles, making it principally possible to set up virtual inter-chip connections as long as the delay sums up to multiples of one cycle.

C. Network Modules

Fig. 2 illustrates the main components of the network module NATHAN as well as the backplane with the connection to the controlling PC needed for user interaction [5]. The central FPGA consists of programmable logic and can be configured to control all connected components. Besides its configurable logic the selected FPGA features embedded MGTs (multi-gigabit transceivers) that are used for the high speed transport network. Furthermore, an embedded IBM PowerPC 405 processor core is available [7], directly interfaced to the programmable logic and capable of running Linux. This provides a convenient way to port the already existing, Linux-based software framework to run on the network modules [8].

The purpose of the NATHAN module is to host an ANN ASIC. Regarding the HAGEN ASIC and its analog synapse operation, this makes analog support circuitry

FPGA	Xilinx Virtex-II Pro XC2VP7: 11k logic cells, 396 user IOs, 8 MGTs, 1 PowerPC (400 MHz)
Memory	1 MB ZBT SRAM up to 1 GB DDR-SDRAM
ANN interface	18 bidirectional LVDS links 46 unidirectional LVDS links HAGEN socket, 2 100 pin SMT connectors for future ANN extension cards
DAC	one 4-channel 12-bit
Interface	100 MHz serial interface for configuration and slow control 8 × Multi-Gigabit Transceivers for inter-module communication
System clock	100MHz/156.25MHz SAW oscillators
Power Supplies	Standard ATX connector 4 × 60A ISRs, 2x1.5V, 2x2.5V
NATHAN manufacturing process	8 Layer Build-Up MicroVia process, min. feature size $100\,\mu m$
NATHAN size	$130\,\mathrm{mm} \times 78\,\mathrm{mm}$
Backplane manufacturing process	4 Layer Standard process, min. feature size $130\,\mu m$
Backplane size	$429\,\mathrm{mm} \times 218\,\mathrm{mm}$

TABLE I
TECHNICAL DETAILS OF THE NETWORK MODULES AND THE BACKPLANE

necessary, like a digital to analog converter (DAC) and a current reference. Together with a temperature sensor these are well confined within a shielded analog area on the network module to minimize digital crosstalk. As the communication interface of the ANN ASIC is kept digital, it is directly connected to the FPGA. In case of the HAGEN chip this is a source-synchronous bus with 16 bidirectional data lines and 3 address lines, all of which follow the LVDS (low voltage differential signaling) standard. The transfer rate of the HAGEN interface adds up to 11.4 Gbit/s. The speed of the FPGA fabric allows to fully exploit the digital bandwidth of the HAGEN chip. Finally, the FPGA has spare differential interconnects available. Together with the HAGEN interconnects these are routed to two extension card connectors which allow for connection to future ANN ASICs with two 16 bit wide differential interfaces providing a bandwidth of up to 25.6 Gbit/s.

Over the ANN ASIC's digital interface the synaptic weights and neural network input/output data is transferred. To process this data and to store software binaries, the PowerPC as well as the rest of the FPGA logic has two memory interfaces available: a DDR-SDRAM and an SRAM bus, each 64 bit wide. The SDRAM architecture allows for large capacities at minimum size and was thus chosen to be the mass storage medium and system memory. One memory socket is integrated to carry a memory module, as commonly used in notebooks, with an addressable capacity of up to 1 Gbyte. The maximum data rate of this interface is 2.5 Gbyte/s and allows constant throughput of network data at the full digital bandwidth of the HAGEN chip.

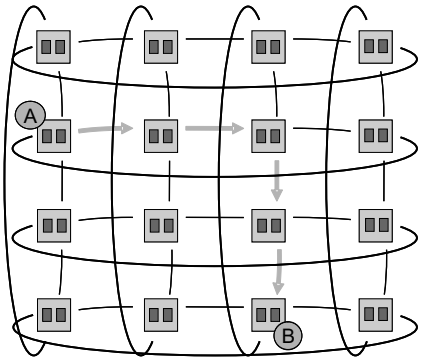


Fig. 4. The transport network forms a hardwired 2D-torus network on the backplane. During operation a worst case routing distance between two network modules A and B of four hops may occur.

A second type of memory allows fast reordering of data, as it is commonly necessary with custom hardware. The architecture of the employed ZBT (zero bus turnaround) SRAMs is well suited for subsequent read and write accesses with no speed penalties. Two SRAM chips with a total capacity of 1 MB are integrated on the network module. Considering a synapse weight to have 8 bit resolution, the whole configuration even of a future megasynapse ANN fits into this memory.

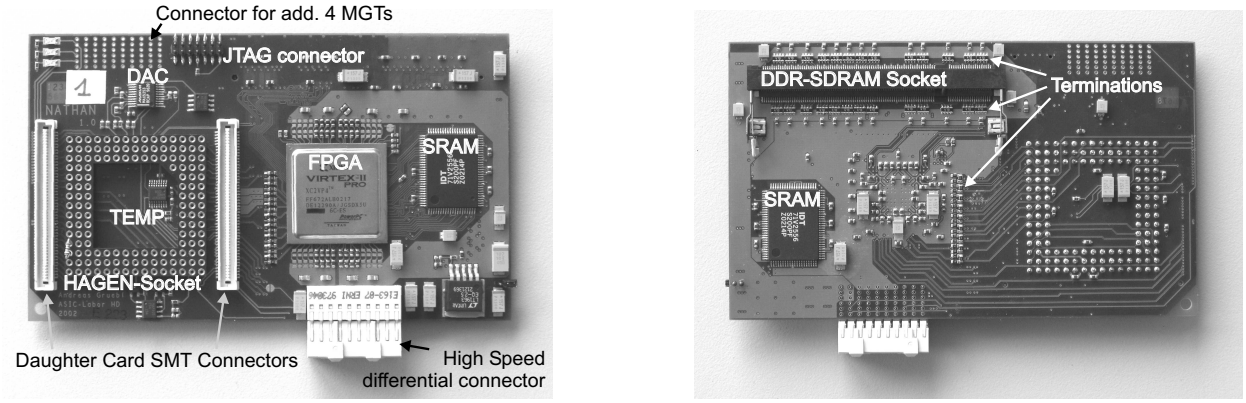
Photographs of the network module NATHAN are given in Fig. 3.

D. Inter-Module Networking

The communication needs between the network modules in a distributed system are met by using the MGTs (multi-gigabit transceivers) that are embedded into the FPGA logic similar to the PowerPC core. The employed FPGA comprises 8 MGTs with a bandwidth of 3.125 Gb/s each. Four of these are routed over a high speed differential connector to the backplane where each network module has hardwired connections to four neighboring modules. Based upon this, 16 network modules on the backplane form a hardwired 2D-torus topology (see Fig. 4). This topology is chosen to minimize passthrough traffic between the modules. The interconnects of the four remaining MGTs are routed to an additional connector at the top edge of the network module. They can be used to construct additional links between the network modules on the same or even other backplanes.

E. Design Challenges

To gain the maximum operating speed on the memory interfaces the large amount of interconnects of these buses have to be routed on the PCB (printed circuit



(a) **(b)**

Fig. 3. Photographs of the presented network module. a) Top view. b) Bottom view.

board) with an equal propagation delay to ensure synchronicity at the respective inputs. The interconnects of the DDR-SDRAM interface in particular have been routed with a maximum relative propagation delay of 15 ps, resulting in a data valid window which allows for the maximum data rate of 2.5 Gbyte/s.

Fast edge rates of the transferred digital signals on both, the network modules and the backplane, require carefully laid out interconnections. Almost all copper traces on the PCBs are implemented as transmission lines with a controlled trace impedance. Those structures particularly require copper planes as an electrical reference potential beneath them to guarantee a controlled impedance [9]. This, in conjunction with the high routing density on the network module required the use of an advanced manufacturing process (see Table I). The signal integrity of critical interconnects was ensured by system-level simulation. Measurements on the most critical MGT path confirmed that we are able to transfer data over 50 cm of standard PCB material and two connectors at a speed of 3.125 Gigabit/s as expected according to the simulations.

III. Functional Description

A. Programmable Hardware

The main reason for the high flexibility of the network module is the programmability of the FPGA. Because all external components and its internal cores (PowerPC CPU and multi gigabit transceivers) are connected to its programmable logic, the operation of the network module can completely be controlled just by configuring the FPGA, which can be reconfigured at any time, if concepts or requirements change.

As mentioned above, the transport gigabit network on the backplane connects all 16 modules in a 2D-torus topology connecting four duplex links to the FPGAs on each module. Because the gigabit transceivers are embedded inside the programmable logic, a routing logic is programmed to be able to pass the network data between different links, which results in a an interconnection of all backplane modules with a worst-case routing distance of four hops (see Fig. 4).

As described in section II-A, the network has to transport two different data types, neural network data and shared memory data, which results in different requirements for the transport process: Concerning neural network data, the transport network has to guarantee constant (but maybe different) latency for the various connections. On the other hand, if shared memory data has to be transported, the network has to be able to process random data requests with minimum latency.

Both requirements can be fulfilled by implementing a low level network layer which processes a continuous stream consisting of packets of all the same size. A connection with the need for constant latency is then implemented by assigning every nth packet of the whole packet stream to belong to that connection, while connections with low latency can be achieved by keeping the overall packet size small. Therefore, the implemented transport network logic is designed to be able to route small packets between several network modules. The router can be configured freely to have maximum flexibility for use with different network models[1].

B. Software

To configure the FPGAs on all modules, they are connected to the controlling PC within a single serial chain. After programming these signal lines are used to implement a 100 MHz token ring-like network to interconnect the PC with all the modules. This allows the PC to have access to each part of the programmable logic inside the FPGA, including the software running on the embedded PowerPCs.

[1]The principles described here are the same as in the ATM network standard

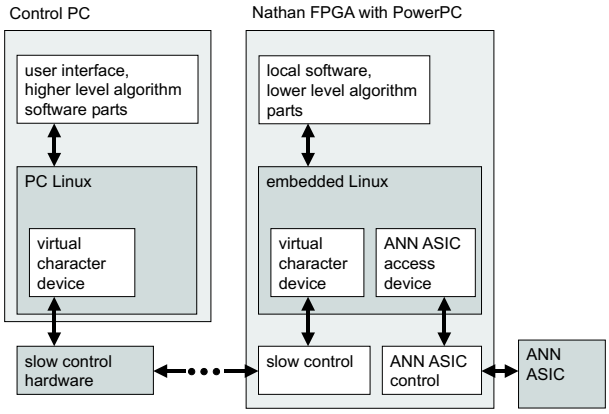


Fig. 5. Schematic view of software interaction between the controlling PC and one example PowerPC

Fig. 5 shows a schematic of how the software is involved in the system. The software framework is split into two parts. The user interface part runs on the controlling PC, while the lower level part which access the ANN ASIC runs on the various local PowerPCs. On them we use a version of the Linux operating system designed for embedded processors [8], which allows to continue using the existing software framework and taking advantage of the features of modern operating systems. For the communication between the controlling PC and the individual PowerPCs the slow control access is presented to Linux as a virtual character device. This allows to set up higher communication protocols using the device, like the PPP protocol to establish the internet protocol TCI/IP, and thus file transfer, login services etc.

C. Operation

During operation user-defined neural network training algorithms are implemented within the existing software framework. The test and training data are sent to the lower level software parts on the several PowerPCs, while the resulting data is sent back to the controlling PC for evaluation.

The presented distributed setup allows to experiment with several possible scenarios of different levels of parallelism: If the networks of interest fit into one ANN ASIC each, the setup can be used to implement several networks in parallel. In this case, no communication takes place, but the setup can utilize the parallelism for higher throughput. If larger neural networks are used, the inputs and outputs of the ASICs could be interconnected as described above to build a large distributed system. If the network has recurrent network connections, the algorithm used has to take care of the different latencies of the network connections. Furthermore, even larger neural networks are possible, in which the size of the network exceeds the total network size of all physically available ASICs. This can be achieved by using the fully digital external representation of the whole neural network state and buffering the remaining parts of it in the available onboard memory. The network operation is then executed in a time multiplexed way.

IV. Example Application

One of the intentions in developing the parallel neural hardware presented in the previous sections was to provide a powerful platform for challenging biological inspired applications. Here, we propose a way to utilize this system to implement the Neocognitron, a massively parallel neural architecture for image understanding, introduced in the early eighties by Fukushima [1]. In this network model, neurons are organized in functionally equivalent layers (Fig. 6(a)). Each layer extracts certain shape-features, as for example edge orientation, from a localized region of the preceding layer and projects the extracted information to the next higher layer. The complexity and abstractness of the detected features grow with the layer height, until complicated objects can be recognized. A layer consists of n feature planes, each of which is assigned to recognize one specific image feature. Neurons belonging to the same plane are identical in the sense that they share the same synaptic weights. This architecture, showing a high degree of self-similarity, seems particularly dedicated to be implemented on a parallel hardware platform.

Fig. 6(b) illustrates our approach to map the Neocognitron's network structure onto the hardware. One HAGEN network block, shown as a gray box, implements one column of n feature neurons, corresponding to the upper part of one of the "chimney" structures in Fig. 6(a). The field of view, consisting of n rectangular regions of the preceding layer, is fed to the block's inputs arranged in a binary linear array. Due to the parallel nature of computing, all n feature neurons can be evaluated in one network cycle. Since within one layer all neurons are column-wise identical, the same network block can be used to process any column of pixels of a given layer. Exploiting the high throughput speed of the HAGEN chip, pixels are fed into the block successively, using attached memory as data buffers. Effectively, one HAGEN block implements a full Neocognitron layer.

Although there are 64 neurons per block available, the possible number of features is at present limited by the input size of one block. Taking for example a field of view of 3×3 pixels, the block's input must be at least of size $n\times3\times3$. Having 128 inputs on each block available, n is limited to 13 features if space for some necessary bias synapses is taken into account. On the other hand, the same design techniques used in developing HAGEN can as well be applied to build ANN chips with an input width of up to 1024 units, so the discussed limitation is not of fundamental nature.

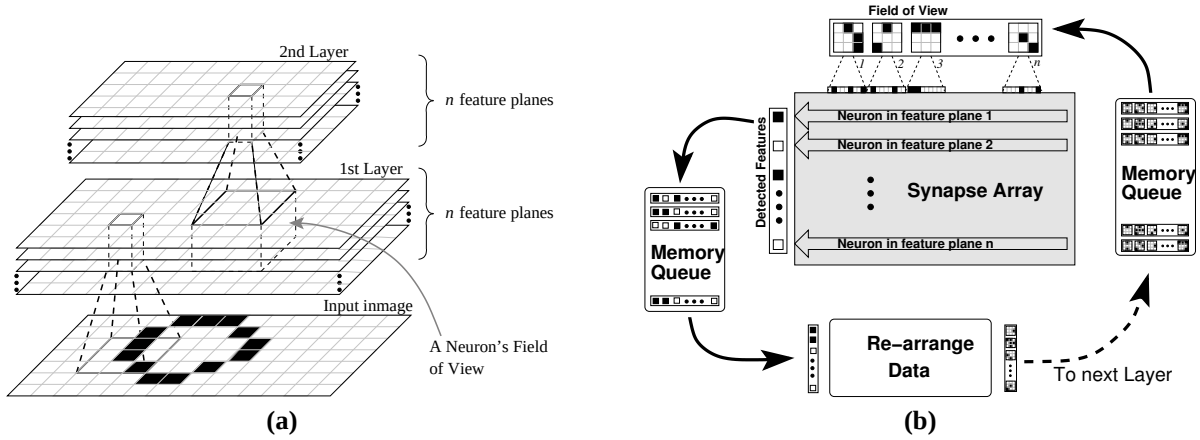


Fig. 6. **(a)** The Neocognitron's hierarchical network. For simplicity, only the first two layers are shown. Every pixel in each feature plane corresponds to one neuron, receiving its input from a localized region of all feature planes of the preceding layer. **(b)** Implementation on the hardware. One column of neurons belonging to the same layer is realized in one network block. All n feature neurons are executed in parallel, each tuned to a specific feature. Between adjacent layers, two memory queues act as data buffers. A special stage is necessary to re-arrange the layer output into field of views serving as input for the next layer.

Unlike the original Neocognitron model which employs a continuous activation function, the used hardware neurons are binary. Principally, it is possible to use several binary neurons in conjunction to simulate quasi-continuous output (see for example [10]). However, we use binary neurons while still obtaining satisfactory results as shown in the experiments below.

Up to now, considerations were restricted to a single network block. In the following, it will be discussed how the possibilities provided by the presented modular hardware architecture can be exploited to allow the operation of multiple chips in parallel. One might suggest to have each ANN chip process a different layer. On the first glance it still does not seem possible to make use of simultaneous computation since a given layer needs the result of its predecessor in oder to operate. However, a closer look reveals that for the current layer to start computing, the outputs of *a portion* of the preceding layer are sufficient. Considering a field of view of size 3×3, the first row of layer m can be evaluated as soon as the results for the first 3 rows of layer $m-1$ are available.[2] Thus, it is possible to set up a chain of ANN modules, each processing a different layer and operating with a time delay equal to the time necessary to compute 3 rows of pixels. In addition, this approach requires a minimum of data buffer size.

Between each layer pair the feature data has to be re-arranged because the upper layer expects its input to be grouped by field of views (cf., Fig. 6). Since each network module carries a fast programmable logic and sufficient memory, these re-arrangements as well as data buffering can be done locally, keeping data transfer and processing time at a minimum.

[2]Indeed, 2 rows plus the first 3 pixels of the 3rd row are sufficient.

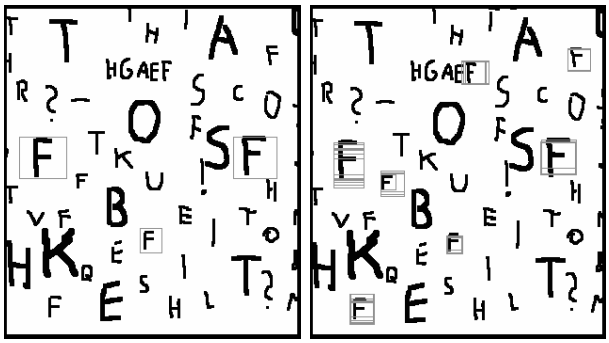


Fig. 7. **Left:** Selections on the input image used to train the "F" feature. **Right:** Occurrence of "F" feature as detected by the trained network.

A. Experimental Results

Preliminary studies have been done on a single HAGEN chip connected to a PC via an FPGA on a PCI card. All layers were executed on that chip consecutively. The Neocognitron model specifies two distinct types of layers. Thus, using two network blocks, one for each layer type, we are able to evaluate the entire network hierarchy without reconfiguring the synaptic weight arrays.[3]

The appropriate weights of the neurons have to be determined in a training procedure. Training is conducted in a supervised manner, successively from simple to complex features. For each feature, one or more example regions in the input image are selected by the human supervisor. By means of size and position of the selected regions, those cells in the hierarchy are determined that are to detect the selected sample features. By a simple

[3]In contrast to the original Neocognitron, we use the same set of features in each layer, to gain scale invariant recognition as proposed in [11].

update rule based on the current input vectors of these cells, a new weight vector is determined and applied to all neurons associated with the trained feature.

A test was conducted on a sample image containing letters with some variances in size and shape. 9 features have been trained using a graphical interface running on the controlling PC: Line segments in each direction (0, 45 90, 135 degrees), line endings (top, bottom), T-junctions (top, left), and the letter "F". In Fig. 7 the training of the most complex feature (letter "F") is shown. At this stage, all lower-level features had already been trained. The left image shows the training selections, as given by the supervisor. On the right, the network result after training is shown. For each "F"-neuron that responded positively, the corresponding region on the input image was determined and marked. Note that due to the self-similarity of the hierarchical network structure the same neuron detects patterns of varying scale. Two "F"s were not detected, but on the other hand, there are no false positives, providing indication of a high specificity of the implemented network model.

Propagating an image with a pixel size of 480x480 trough 13 layers, the computing time of the HAGEN chip including reading and writing the data from and to local memory on the PCI board, was only 730 ms. Note that in this time approximately 12 million neurons with each receiving 100 inputs (90 input + 10 bias) were evaluated. This time could be easily decreased by a factor of 2 by employing the two unused blocks of the chip processing different image regions in parallel. At present, further overhead is introduced by data transfer and computation time, since the re-arrangement stage was executed on the controlling PC. We are convinced that once this stage is implemented in the local FPGA, the additional time required will be of the same order of magnitude as the time for executing the neural network. In conjunction with parallel execution, as discussed above, high resolution images can then be processed with a rate of several frames per second.

V. Conclusions & Outlook

We presented autonomous network modules, each providing a complete operating environment for a neural network ASIC. The neural networks as well as their controlling software can be distributed over several modules working in parallel. At present, the system is set up to operate the mixed-mode analog neural network ASIC HAGEN whose network model allows the extension across chip boundaries if isochronous communication is ensured. This requirement is met with a transport network which physically uses multi-gigabit transceivers to transport the neuron communication. This transport network is also used to implement a shared memory architecture distributed across the modules. Finally, we presented several scenarios how the massive parallelism of this system can be exploited.

The hardware is developed within a European project, called SenseMaker, and provides the basis for the exploration of sensory fusion by an biologically inspired artificial system. The presented adoption of the Neocognitron shows how a higher-level model can be implemented on the system. The system design allows to replace the used ANN ASIC with future versions while reusing the parallel infrastructure. This will also allow to research lower-level biological principles, e.g. using spiking neuron models.

VI. Acknowledgement

This work is supported in part by the European Union under the grant no. IST-2001-34712 (Sensemaker).

References

[1] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119–139, 1988.

[2] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, pp. 1019–1025, September 1999.

[3] SenseMaker, "A multi-sensory, task-specific adaptable perception system." EU Contract No. IST-2001-34712.

[4] J. Schemmel, S. Hohmann, K. Meier, and F. Schürmann, "A mixed-mode analog neural network using current-steering synapses," *Analog Integrated Circuits and Signal Processing*, vol. 38, pp. 233–244, February-March 2004.

[5] A. Grübl, "Eine FPGA-basierte Plattform für neuronale Netze." Diploma Thesis, Heidelberg University, HD-KIP-03-02, 2003.

[6] J. Hennessy and D. Patterson, *Computer Architecture - A Quantitative Approach*. San Francisco, California: Morgan Kaufmann Publishers, Inc., 1995.

[7] Xilinx, Inc., www.xilinx.com, *Virtex-II Pro Platform FPGA Handbook*, 2002.

[8] A. Sinsel, "Linuxportierung auf einen eingebetteten PowerPC 405 zur Steuerung eines neuronalen Netzwerks." Diploma Thesis, Heidelberg University, HD-KIP-03-14, 2003.

[9] S. Hall, G. Hall, and J. McCall, *High-Speed Digital System Design*. New York, Weinheim: John Wiley & Sons, Inc., 2000.

[10] F. Schürmann, S. Hohmann, K. Meier, and J. Schemmel, "Interfacing binary networks to multi-valued signals," in *Supplementary Proceedings of the Joint International Conference ICANN/ICONIP 2003* (O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, eds.), pp. 430–433, 2003.

[11] J. Teichert and R. Malaka, "An association architecture for the detection of objects with changing topologies," *Proceedings of the International Joint Conference on Neural Networks (IJCNN) 2003*, pp. 125–130, 2003.