NeuroPipe-Chip: A Digital Neuro-Processor for Spiking Neural Networks

Tim Schoenauer, Sahin Atasoy, Nasser Mehrtash, and Heinrich Klar, *Member, IEEE*

***Abstract*—Computing complex spiking artificial neural networks (SANNs) on conventional hardware platforms is far from reaching real-time requirements. Therefore we propose a neuro-processor, called NeuroPipe-Chip, as part of an accelerator board. In this paper, we introduce two new concepts on chip-level to speed up the computation of SANNs. These concepts are implemented in a prototype of the NeuroPipe-Chip. We present the hardware structure of the prototype and evaluate its performance in a system simulation based on a hardware description language (HDL). For the computation of a simple SANN for image segmentation, the NeuroPipe-Chip operating at 100 MHz shows an improvement of more than two orders of magnitude compared to an Alpha 500 MHz workstation and approaches real-time requirements for the computation of SANNs in the order of 10^6 neurons. Hence, such an accelerator would allow for applications of complex SANNs to solve real-world tasks like real-time image processing. The NeuroPipe-Chip has been fabricated in an Alcatel 0.35–μm digital CMOS technology.**

***Index Terms*—Neuroaccelerator, neurochip, pulse-coded neural networks, spiking neural networks.**

I. Introduction

SPIKING (or pulse-coded or pulsed) neurons represent complex integrate-and-fire neurons. Synchronized firing of neuronal assemblies could serve the brain as a code for feature binding, pattern segmentation, and figure/ground separation [10], [3]. Spiking artificial neural networks (SANNs) are able to model such synchronization since they take into account the precise timing of spike events. They are therefore subject of investigations for biology-inspired image processing applications [16], [9]. However, employing SANNs for image processing requires complex networks in the order of 10^6 of spiking neurons [12]. Computing large networks of complex neuron models is a computational expensive task and leads to long run times even for high-performance workstations [7]. Furthermore, to solve real-world tasks there is a need for computing complex networks in real-time, which can only be achieved by supercomputers or dedicated hardware. The NeuroPipe-Chip is part of such a dedicated digital hardware accelerator system. For several reasons we chose a digital and not an analog implementation. Analog hardware offers a potential for compact, low-power realizations by taking advantage of the inherent characteristics of silicon devices. An example for an analog time-multiplexed accelerator for SANNs is described in [19] and proposals exist to link several analog chips to a neuromorphic system [20]. However, the accuracy of analog signal processing is limited, memory storage of analog signals is area hungry or volatile and transmitting them over chip boundaries is difficult. Since we aimed at the computation of very complex networks with a fairly high accuracy (e.g., 16 bits) with extensive programming capability, the digital approach was first choice.

Main objective of the paper is to present the architecture of the NeuroPipe-Chip employing two new concepts for a better computational performance. Also HDL-based simulation results of the accelerator system are described which allow to evaluate the NeuroPipe-Chip performance. For a better understanding, in Section II and III of this paper general aspects of SANNs, the simulation of SANNs and a review of our accelerator system is given.

Manuscript received November 30, 2000. Prototypes of the NeuroPipe-Chip have been fabricated and funded in part by the EUROPRACTICE service, funded by the IST program of the European Commission.

The authors are with the Institute of Microelectronics, Technical University of Berlin, D-10587 Berlin, Germany (e-mail: tim.schoenauer@multilink.de, sahin.asatoy@bln.siemens.de, nasser@mikro.ee.tu-berlin.de, klar@mikro.ee.tu-berlin.de).

Publisher Item Identifier S 1045-9227(02)00344-2.

II. Computing Spiking Neural Networks

Spiking neurons represent biophysical models that account for properties of real neurons without descending to the level of ionic currents. They rather model the integrated signal flow of incoming action potentials through parts of the neuron, in particular the synapses, dendrites, soma and axon. Opposed to rate-coded models, spiking neuron models encode their information in the exact timing of a neuron's firing event not in the frequency rate of firing. Therefore interneuronal communication takes place solely via action potentials (also called spikes or pulses). A discrete-time model of a generic spiking neuron with q feeding dendrites (excitatory or inhibitory), modulated multiplicatively by a linking dendrite and a dynamic threshold is shown in Fig. 1. Each dendrite as well as the dynamic threshold is modeled by a leaky integrator. There are various types of dendrites with different kinds of influence on the calculation of the membrane potential. Feeding dendrites may be modulated by linking dendrites for a better synchronization of neuronal assemblies as proposed by Eckhorn *et al.* [4], while other dendrites are not influenced by linking and may serve as inhibitory dendrites. Furthermore, delay elements at the output of the model represent axonal delays.

For the design of our accelerator system, we assume SANNs to be locally and therefore sparsely connected. The networks are composed of several layers where each layer consists of neurons with equal properties. Neurons may be connected laterally (within one layer) as well as to other layers. Connections are

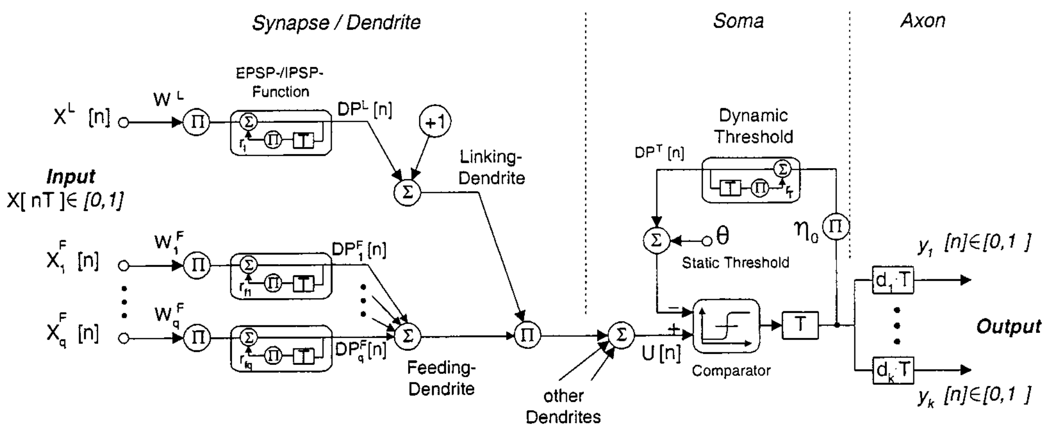

Fig. 1. Discrete-time representation of a generic spiking neuron model [4].

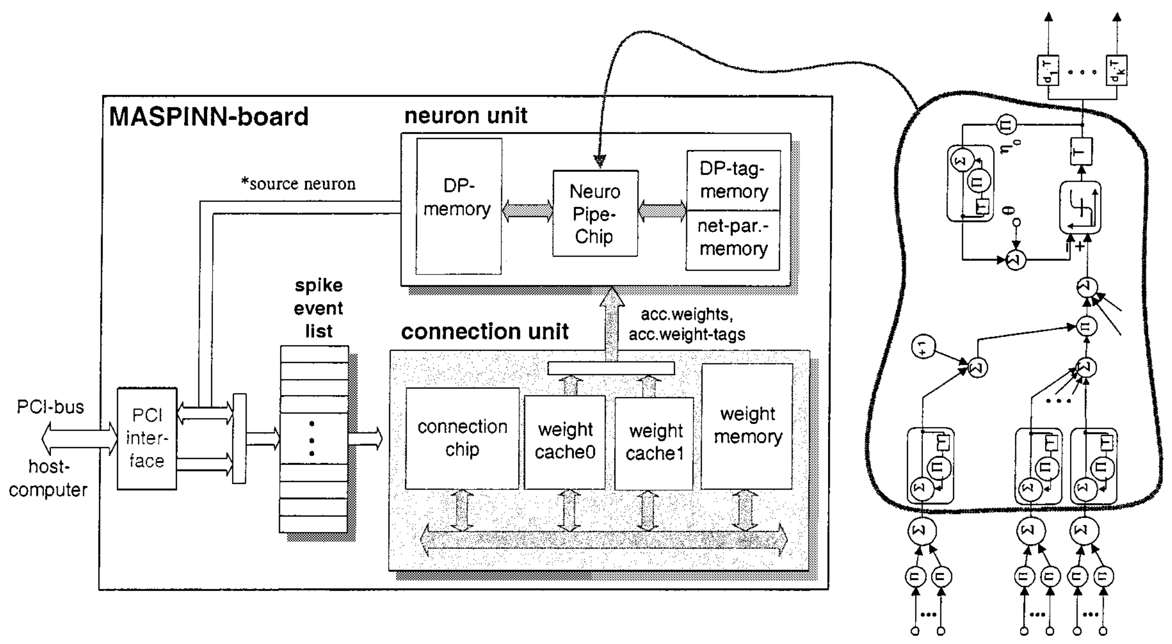

Fig. 2. Basic structure of a MASPINN-board. Right: circled part is computed by the NeuroPipe-Chip.

usually structured in receptive fields. The network activity resembles the average number of spikes per time slot divided by the total number of neurons. Network activity of SANNs is typically low i.e., <1%. The computation of SANNs by digital hardware is performed in discrete time. For real-time computation, a time interval (which we refer to as time slot) of 1 ms is considered sufficient for the entire network data to be updated [1].

III. Accelerator System for Spiking Neural Networks: MASPINN

In order to achieve real-time computation of very complex SANNs we proposed an accelerator system called memory optimized accelerator for spiking neural networks (MASPINN), which is an accelerator board connected to a host computer via a PCI-bus [13]. As shown in Fig. 2, an MASPINN-board consists of three main units: a neuron unit, which computes the neuron model, a spike event list, which stores the addresses of source neurons (neurons emitting a spike) and a connection unit, which determines target neurons (neurons receiving a spike) and corresponding weight values.

The MASPINN-structure is based on previously proposed concepts such as a spike event list, a sender-oriented connectivity list and tagging of dendrite potentials [5], [6], which are summarized as concepts 1–3 in Table I. We refer to dendrite potentials (DPs) as the state values in the leaky integrators of the neuron model (Fig. 1). Two additional concepts are associated with the MASPINN-architecture: weight caching and a compressed DP-memory (concepts 4–5 in Table I), which will be discussed more thoroughly in the following. In principle, all of the previously mentioned concepts either try to reduce the amount of computation to a minimum or to make the required computation more suitable for a hardware realization.

1) Compressed DP-Memory: Due to a low network activity, only part of the DPs in a SANN receive an input. Therefore many DPs decay to zero and have no influence on the membrane potential of a neuron. Hence, these DPs do not need to be accessed or processed. This can be achieved by using tags which

TABLE I
CONCEPTS OF ACCELERATORS FOR COMPUTING SANNS

Concept	Principle	Benefit
1.Spike event list	addresses of neurons emitting a spike are stored in an event list during one time slot and considered as source neurons during the next time slot.	* efficient communication between time slots
2.Sender-oriented connectivity list	connections are stored in a list which contains for each address of a source neuron the addresses of all connected neurons (target neurons) and a weight value characterizing the strength of the connection.	* efficient memory organization to store network connectivity of an arbitrary topology
3.Tagging of dendrite potentials (DPs)	DPs are tagged to be valid or irrelevant. If they are not valid they are neither accessed nor processed.	* reduced IO-requirements * reduced computational load
4.Weight caching	weighted spikes are accumulated in weight caches within the connection unit one time slot ahead before being propagated to the corresponding DP in the neuron unit	* parallelization of main processing steps * reduced IO-bandwidth
5.Compressed DP-memory	only valid DPs are stored and accessed; each word read from the DP-memory contains only valid DPs; their address is generated by analyzing the stream of tag-bits.	* efficient use of IO-bandwidth between neuron processor and DP-memory

mark the relevance of a DP with a single bit. If the value of a DP drops below a user-defined threshold its tag-bit is set from "1" to "0" (concept 3 in Table I).

Normally there is a place in memory reserved for every DP, since the relevance of a DP might change during the computation. This leads to inefficient access of DP-data once data-words which are several DPs wide are stored in a single memory. These words might inevitably contain irrelevant DPs and cause a loss of effective bandwidth between the DP-memory and the neuron processor. What we refer to as compressed DP-memory represents a solution to that problem: only the relevant DPs are stored in memory in consecutive order (first neuron in first layer to last neuron in last layer). That way only relevant DPs are stored and transferred between memory and processor.

A disadvantage of this approach is the loss of a direct relationship between the physical memory-address of a DP and its logical address in the computed network. However this problem can be solved by analyzing the stream of tag-bits while consecutively processing all DPs during each time slot. Each tag-bit must be counted and at each value "1" the counter value represents an address belonging to one of the DPs in the consecutive stream of DPs, which is read in parallel. This task is performed by on-chip tag-to-address- and address-to-tag-converter of the NeuroPipe-Chip (Fig. 3).

2) Weight Caching: Weight caches represent copies of a complete DP-memory (for each DP there is a certain place in memory reserved). They are used to accumulate all weighted spikes occurring in one time slot. The accumulated weights then serve as target neuron input during the next time slot. Two weight caches are required: during an entire time slot one functions as an accumulator while the other one sends weights, which have been accumulated in the previous time slot, to the neuron unit. The function (accumulator/sender) of the weight caches alternates with each new time slot. Since thereby the complete stimulation data for each DP is known already at the beginning of the next time slot, the main processing steps to compute the neuron model may be processed in parallel by a pipelined datapath (see Fig. 3). The pipelined processing will be outlined in more detail in the following section.

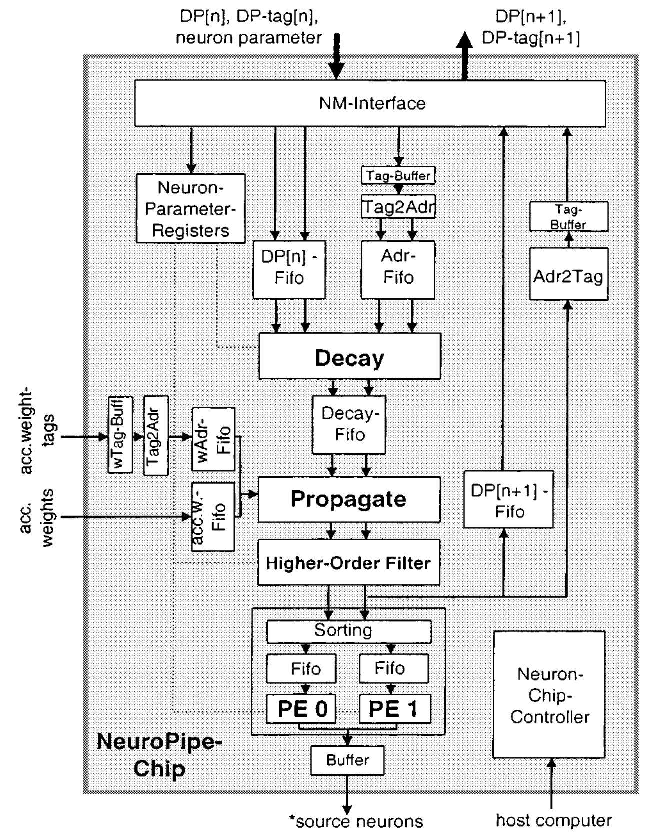

Fig. 3. Basic blocks of the NeuroPipe-Chip.

IV. ARCHITECTURE OF THE NEUROPIPE-CHIP

Main task of the NeuroPipe-Chip is to compute a spiking neuron model as depicted in the circled part of Fig. 2. Dendrites may also be modeled by higher order filters (not shown in Fig. 2). In order to allow different kinds of neuron models to be computed, the NeuroPipe-Chip computes a programmable neuron model: with a program code the user specifies the number of DPs per neuron and how each DP contributes to the membrane potential (e.g., excitatory, inhibitory, multiplicatively). All neurons of one layer are supposed to have the same characteristics.

For each time slot the computation of the neuron model contains the following sequence of steps [8].

Decay: DPs are decayed ("leaky integration").

Propagate: Stimuli from other neurons in the network or the input layer ("accumulated weights") are added to the corresponding DP.

Output: DPs are combined to a membrane potential according to a program code. When exceeding a threshold the neuron spikes and therefore its address is written to the spike event list.

The basic computational steps (decay, propagate, output) have been implemented in the dataflow architecture of the NeuroPipe-Processor as depicted in a block diagram of the NeuroPipe-Processor in Fig. 3. Therefore the basic operations to compute a neuron model are hard-wired. However, by specifying, e.g., the number of DPs per neuron and selecting different functionality of each DP with a program code, the user may program the NeuroPipe-Processor to compute various neuron models.

For example, to compute the neuron model of the benchmark network we used in Fig. 6(a), (1)–(4) need to be solved to update the DPs in time slot $[n]$, which are DP^F, DP^L, DP^I and DP^T

$$DP^F[n] = w^F \cdot x^F[n] + r_F \cdot DP^F[n-1] \tag{1}$$

$$DP^L[n] = \sum_{j=1}^{p} w_j^L \cdot x_j^L[n] + r_L \cdot DP^L[n-1] \tag{2}$$

$$DP^I[n] = \sum_{j=1}^{q} w_j^I \cdot x_j^I[n] + r_I \cdot DP^I[n-1] \tag{3}$$

$$DP^T[n] = \eta_0 \cdot y[n] + r_T \cdot DP^T[n-1]. \tag{4}$$

The upper index refers to the feeding, linking, inhibitory, and the dynamic threshold character of the parameter, where x is a neuron input and y the neuron output, w a connection weight, and r a decay factor. During the decay phase the decay term $r \cdot DP[n-1]$ is calculated. In the propagate phase, the accumulated weight value $\sum w \cdot x[n]$ is added to the decay term. Based on the updated DPs, the membrane potential $u[n]$ is calculated by (5) and upon spike emission is determined by (6) during the output phase

$$u[n] = DP^F[n] \cdot (1 + DP^L[n]) - DP^I[n] \tag{5}$$

$$y[n] = \begin{cases} 1, & \text{if } (u[n] \geq DP^T[n] + \Theta) \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

While η_0 in (4) represents the strength of the negative feedback by the dynamic threshold potential, Θ refers to the static threshold offset.

Thanks to the concept of weight caching, the complete set of data to compute the neuron model is available at the beginning of a new time slot: all DPs and accumulated weights. Therefore, starting with the first relevant DP of the first layer, DPs are consecutively fed into a datapath pipeline performing the steps: decay, propagate, and output (see Fig. 3). In addition to a fully pipelined datapath, several such datapaths in parallel achieve a further speed up. The NeuroPipe-Chip was designed with two datapaths in parallel.

Per clock cycle each of the two parallel datapaths receives a relevant DP from the DP-memory via the neuron memory-(NM)-Interface (see Fig. 3). The neuron memory encloses the DP-, the DP-tag- and the neuron-parameter memory. From the DP-tag-stream, the corresponding DP-address is generated by a tag-to-address-converter. DP-data and -address now enter a pipeline stage where the DP-data is multiplied with a decay factor r ("decay stage"). If the result is below a user-defined threshold, the DP is regarded irrelevant and the DP is removed from further processing. In the next pipeline stage an accumulated weight from the weight cache is combined with the DP ("propagate stage"): An accumulated weight and a DP with an equal address are summed; an accumulated weight with no corresponding DP becomes a new DP and is inserted into the DP-stream so that the consecutive order of DPs is remained. Since DP-removal in the decay stage and DP-insertion in the propagate stage might cause stalls within the datapath pipeline, first-in–first-out-(FIFO)-Memories have been added in front and behind the pipe stage to buffer data irregularities.

At the output of the propagate stage the DP is now completely updated and a copy of the data is written back to the DP-memory. The corresponding DP-address is converted back into a "1" at the proper place in the DP-tag-stream by an address-to-tag-converter. From the propagate stage a copy of the DP is also delivered to a subsequent pipe stage, the higher order filter stage, where DPs are cascaded to model higher order filter functions. A sorting stage then maps all DPs of the same neuron to one of two parallel processor-elements (PEs). A PE computes the membrane potential and determines the spike activity of the neuron. Optimal load balancing of the parallel PEs is achieved by the sorting stage. It uses the half-full-flags of the FIFOs in front of each PE to decide which PE the next neuron will be mapped to. In case a neuron is active, its address is written to the spike event list.

V. Novel Concepts of the NeuroPipe-Chip

The architecture of the NeuroPipe-Chip was designed to take advantage of the MASPINN-concepts such as weight caching and a compressed DP-memory. However, also two novel concepts in the NeuroPipe design allow further increase in system performance: an on-chip inhibition unit and data preanalysis.

1) On-Chip Inhibition Unit: In SANNs for image processing, inhibition is commonly used to control network activity and to generate a winner-take-all mechanism: e.g., to separate objects in time by an SANN for image segmentation [11]. Such an inhibition module receives spikes from all neurons or a large portion of the network. It then applies equally distributed negative feedback to these neurons. Typically, the connections to and from the inhibition module have similar strength. Therefore, an inhibition module may be implemented conveniently on-chip: only a few parameters are required for characterization; also, by placing the on-chip inhibition unit

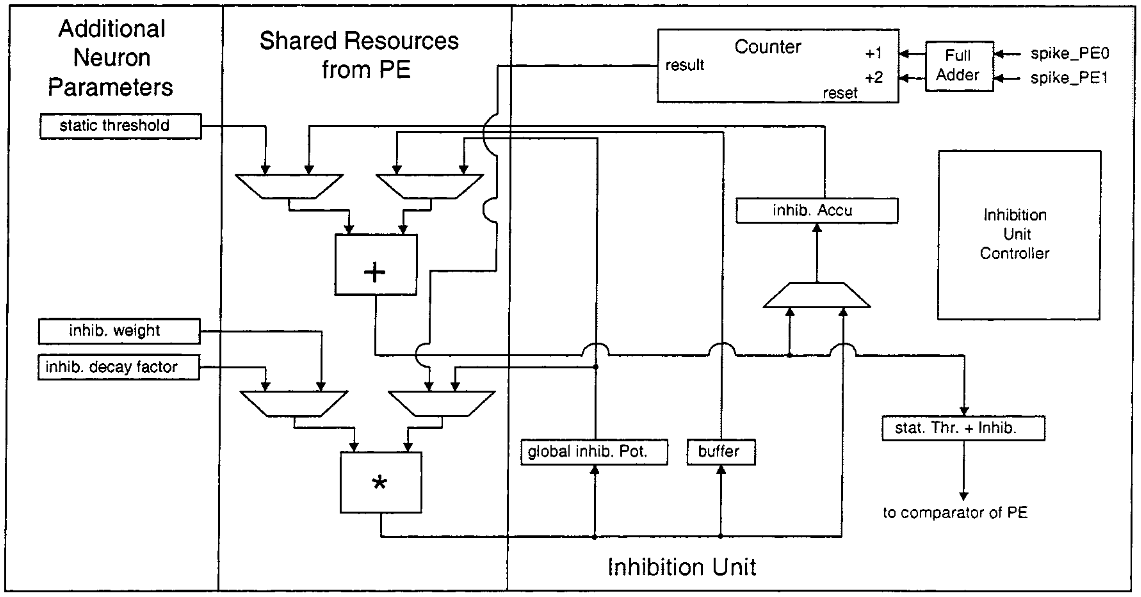

Fig. 4. On-chip inhibition unit.

within a processor element, resource sharing of arithmetic modules, like multipliers, minimizes area overhead.

The basic structure of the on-chip inhibition unit is shown in Fig. 4. The left part of Fig. 4 shows registers with additional neuron parameters required. The middle part represents the required arithmetic elements. Since the computation takes only a few clock cycles and is required only once during a time slot, the arithmetic elements of a PE may be used. The right part shows the main elements of the on-chip inhibition unit: only a counter, registers for the inhibition data, and a controller are necessary.

The global inhibition unit works as follows. Each spike of a neuron within a layer increments a counter. At the end of layer processing, the counter value is multiplied with an inhibition weight, buffered, and added to an inhibitory accumulator and finally the counter is reset (see Fig. 4). These actions repeat for each layer of the network during one time slot. Hence, at the end of a time slot the value in the inhibitory accumulator represents some equivalent of the network activity during this time slot. It may be used during the next time slot to inhibit the network. For example the accumulated value is added to a so-called global inhibitory potential DP^I_{glob}, which is decayed each time slot by an inhibitory decay factor r_I. Considering a simple case of uniform global inhibition, each neuron of the network is an element of the set N_{inhib} and, if active, it takes part in the inhibition process with a synaptic strength w^I_{glob}. The inhibitory DP (DP^I) of (3) previously had to be computed for each neuron. Now it may be substituted by a global inhibitory potential DP^I_{glob} which is computed only once per time slot for the entire network

$$DP^I_{\text{glob}}[n] = w^I_{\text{glob}} \cdot \sum_{j=1}^{j \in N_{\text{inhib}}} y^I_j[n-1] + r_I \cdot DP^I_{\text{glob}}[n-1]. \quad (7)$$

By computing the inhibitory potential in the on-chip inhibition unit only once per time slot instead of computing an inhibitory DP for each neuron of the network, memory, IO-bandwidth, and computation time are saved.

Since a global inhibitory potential is identical for all neurons, it is not necessary to take it into account for each neuron of a layer during membrane potential calculation. Instead, it may be computed once at the beginning of layer processing and added to the static threshold (see Fig. 4) modifying (6) to

$$y[n] = \begin{cases} 1, & \text{if } (u[n] \geq DP^T[n] + \theta + DP^I_{\text{glob}} \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

In the case that the inhibition is not global but only local to a few layers, several inhibition units are required. In the NeuroPipe-Chip, we implemented two on-chip inhibition units, one in each PE.

2) Data Preanalysis: Computational resources can be furthermore saved by analyzing which DPs need to be taken into account for membrane potential calculation. A reduction of computational load is particularly important when considering a multiplication of 16 bit operands (e.g., DP) as required in a PE for example during a linking multiplication. Such a multiplication may demand several clock cycles in a hardware realization. The multiplier in the PE of the NeuroPipe-Chip was designed as a two-stage pipelined booth-encoded-Wallace-tree-multiplier in order to achieve a clock frequency of 100 MHz in the 0.35-μm-CMOS technology. Therefore, during each multiplication the PE is busy for three clock cycles. However, the rest of the datapath stages (e.g., decay stage, propagate stage) is designed to compute an output within one clock cycle. Thus, the output stage of the NeuroPipe-Processor with PEs performing linking operations might become a bottleneck and could require the introduction of wait cycles in other processing stages of the chip.

Computational load of the PE can be reduced by analyzing the character and validity of DPs belonging to a certain neuron.

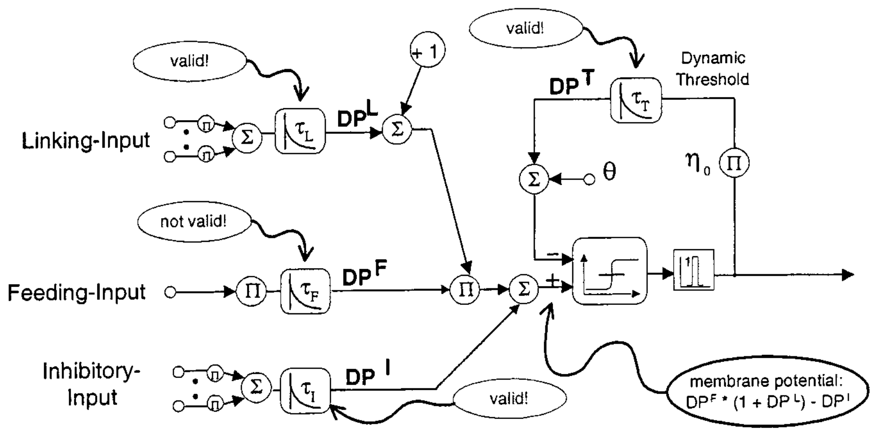

Fig. 5. Example scenario where data preanalysis detects unnecessary computation of a neuron.

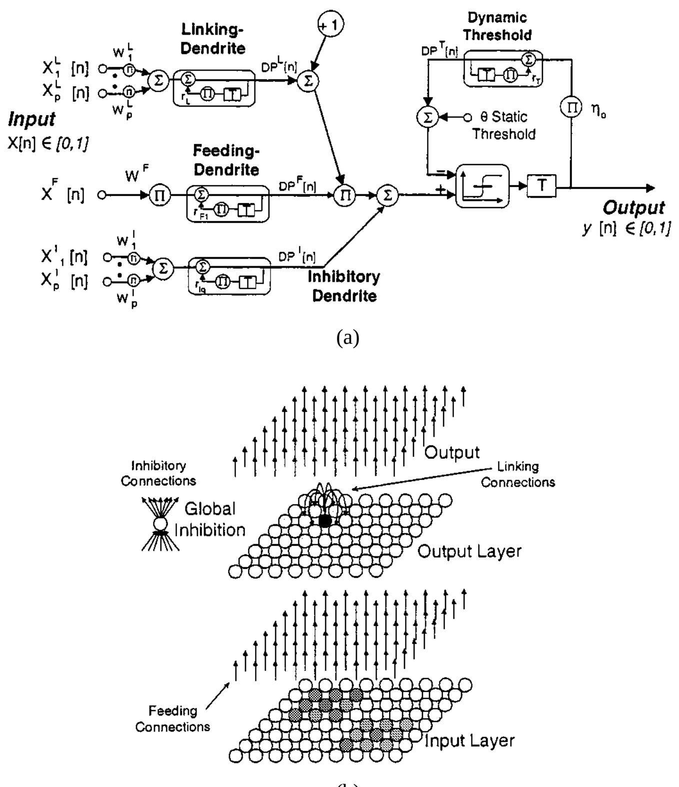

Fig. 6. (a) Neuron model. (b) Network topology [11].

For example, if there are no valid DPs of a neuron with additive (excitatory) influence on the membrane potential, it does not make sense to calculate the ones with multiplicative or subtractive (inhibitory) influence even if they are valid: in any case the neuron will not emit a spike. An example of such a scenario based on the neuron model in Fig. 6(a) is depicted in Fig. 5.

In the NeuroPipe-Architecture, the sorting unit keeps track of the types of DPs in the datapath pipeline (see Fig. 3). First of all the sorting unit maps all DPs belonging to one neuron to a certain PE. To achieve an optimal load balancing between the two PE, the sorting unit takes into account the fill state of the two PE-FIFOs. Also, the data preanalysis is performed. In case data preanalysis reveals that no additive DPs have occurred and will occur for one neuron, all multiplicative or subtractive DPs for this neuron are neglected. Thereby the work load of the PEs is reduced.

VI. System Simulation Including a Model of the NeuroPipe-Chip

In the process of designing the NeuroPipe-Chip, we realized a HDL-model on register transfer level (RTL) of the chip in the hardware description language VHDL. In the design flow of an ASIC (Application Specific Integrated Circuit), the register transfer level of a chip represents an intermediate stage between an algorithmic description and the actual circuit layout. The correspondence of the RTL-model of the NeuroPipe-Chip to the fabricated prototype in respect to timing and functionality was verified during chip test (see Section VII). However, the entire MASPINN-system has not yet been implemented in hardware. Therefore, for performance evaluations the MASPINN-system is modeled in behavioral (algorithmic) VHDL-code. The algorithmic MASPINN-model is used as a testbench for the NeuroPipe- RTL-model. The MASPINN-model always provides/receives the required data to/from the NeuroPipe-Chip within one clock cycle. This is a realistic assumption for the interfaces to the spike-event list and the memories within the neuron unit where fast static random access memories (SRAMs) may be used. However, for the interface to the connection unit, this assumption is only valid up to a maximum number of connections. Exceeding such a number of connections, the connection unit will become a bottle neck for computation speed. In that case the access of connection weights and their accumulation in the weight caches will take more time than the computation of the neuron model by the NeuroPipe-Chip. Building a MASPINN-board with state-of-the-art components, up to 50–100 connections per neuron at a network activity of about 0.5% seem feasible.

In order to validate the functioning of the NeuroPipe-Chip and to evaluate its performance, we chose a simple SANN for image processing [11] as a benchmark. This benchmark network has been selected for two reasons. On the one hand, it provides typical characteristics of SANNs for image processing which

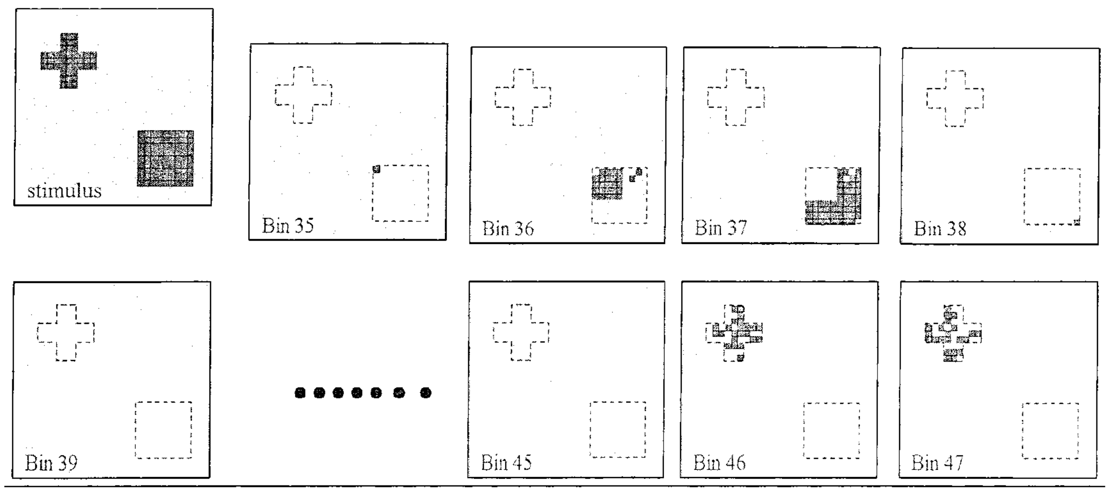

Fig. 7. Results of an MASPINN system simulation using a simple benchmark network (1000 neurons) [11].

TABLE II

PERFORMANCE OF THE NEUROPIPE-CHIP WITH AND WITHOUT NEW CONCEPTS (ON-CHIP INHIBITION, PREANALYSIS) AND IN COMPARISON TO OTHER HARDWARE PLATFORMS (* MEASURED, # SIMULATED, + EXTRAPOLATED, o ESTIMATED). LISTED ARE THE TIME INTERVALS IT TAKES HARDWARE PLATFORMS TO COMPUTE ONE TIME SLOT OF A SANN BENCHMARK [11] (NEURON WITH 4 DP, NETWORK ACTIVITY ABOUT 0.4%, RELEVANT DP ABOUT 12% EXCL. INHIB. DP)

Number of Neurons	**Workstation Alpha** 500MHz	**SPIKE128K** 10MHz FPGA-based (4 boards)	**ParSpike** 100MHz 64 DSPs	**MASPINN with NeuroPipe-Chip** 100MHz (preanalysis **off** & on-chip inhib. **off**)	**MASPINN with NeuroPipe-Chip** 100MHz (preanalysis **on** & on-chip inhib. **on**)
1K	0.56ms^o	<< 1ms^o	<< 1ms^o	0.012ms$^\#$	6.5μs$^\#$
128K	67ms*	10 ms*	1ms^o	1.5ms$^+$	0.83ms$^+$
1M	650ms*	-	8ms^o	11.8ms$^+$	6.5ms$^+$

are also inherent to SANNs solving more elaborated tasks than shown in Fig. 7 (e.g., as [16]). On the other hand, the chosen network is simple to implement in C- and VHDL-code and yields reasonable simulation run times.

As shown in Fig. 6(b), the network consists of an output layer and a global inhibition neuron. The neuron model of the benchmark network is shown in Fig. 6(a). It has three types of inputs: a feeding, a linking and an inhibitory input. The refractory period of the neuron is modeled by an additional leaky integrator referred to as dynamic threshold. The input to the network is given by a binary image where each pixel is associated with the feeding dendrite of an output layer neuron. Besides these feeding connections there are linking connections organized in receptive fields. They connect neurons of the output layer laterally with their 9×9 nearest neighbors. A global inhibition neuron receives input from all output layer neurons. Its output is connected to the inhibitory dendrite of all output layer neurons.

Fig. 7 shows the result of a VHDL-based system simulation run for a network of 1000 neurons and two input stimulus objects: a "plus" and a "square." The SANN performs an image segmentation: neurons associated to the "plus" spike during a certain time interval (bin35–bin38) while neurons associated with the "square" spike with a phase shift in another time interval (bin46–bin47). For a typical simulation run, the network activity was 0.4% and 12% of the DPs were relevant excluding inhibitory DPs.

As a measure of computational performance, the time to compute the new state of the network (time slot) is given in Table II for different accelerator architectures and network complexities ranging from about 1000 to 1 000 000 spiking neurons. As suggested by the VHDL-based system simulation, the NeuroPipe-Chip at 100 MHz (with the architectural features on-chip inhibition unit and preanalysis) computes a time slot for a benchmark network of one million neurons in about 6.5 ms. The NeuroPipe-Chip with on-chip inhibition unit and preanalysis thereby yields an improved performance of a factor of 1.8 compared to the NeuroPipe-Chip without these concepts.

Table II also attempts to compare the performance of the NeuroPipe-Chip to other hardware platforms. The purpose of Table II is to give a coarse overview of the performance of different accelerator approaches. It is not meant to be an accurate comparison between these platforms, since most of the values in Table II are based on estimation. Out of several other accelerators architectures for SANN, which have been proposed, two have been chosen for comparison: the SPIKE128K-and the ParSpike-accelerator. The SPIKE128K [17] represents a field-programmable gate array (FPGA)-based approach. ParSpike [18] connects 64 commercial digital signal processors (DSPs) with programmable logic and memory to an accelerator architecture optimized for the computation of SANN.

Also listed in Table II is the execution time of an Alpha workstation to compute the benchmark network coded in C. Executing the C-code on a 500 MHz Alpha workstation took 650 ms (averaged over 1000 time slots) to compute one time slot of a SANN of one million neurons. Thereby real-time requirements are missed by almost three orders of magnitude. The SPIKE128K was designed and built at the Technical University of Paderborn [17]. It consists of four boards of programmable logic and exhibits a performance improvement compared to an Alpha workstation of almost a factor of seven for a network of about 128 K spiking neurons. The ParSpike-accelerator with 64 DSPs (AD21160 from Analog Devices) running at 100 MHz promises another order of magnitude performance improvement compared to the SPIKE128K and suggests about the same performance as the NeuroPipe-Chip embedded in the MASPINN-system. The computational speed of the NeuroPipe-Chip within the MASPINN-system for the benchmark SANN of 10^6 neurons is still a factor of about 6.5 slower than the real-time requirement of 1 ms per time-slot demands. However, several MASPINN-boards in parallel and/or a NeuroPipe-Chip with more parallel datapaths would multiply the performance of the MASPINN-system. By designing a NeuroPipe-Processor with four parallel datapaths (instead of two of the prototype) and running four MASPINN-boards in parallel, e.g., real-time computation of SANNs in the order of 10^6 neurons could become feasible.

VII. Design of the NeuroPipe-Chip

The NeuroPipe-Chip has been implemented as a digital standard-cell design in an Alcatel five-metal layer 0.35 μm-CMOS-process. For the implementation of the NeuroPipe-Chip, a design flow has been developed combining various CAD-tools. For design entry on register transfer level in graphical VHDL the tool *Visual-HDL* of *Summit Design* was used [14]. A pure textual VHDL-code is generated by *Visual-HDL* and fed into a logic-synthesizer. We used *Design-Compiler* of *Synopsys* to optimize the RTL-design representation and map it to a standard-cell library [15]. The layout of the circuit was generated by *Silicon-Ensemble* of *Cadence* [2]. Even though the architecture of the NeuroPipe-Chip takes advantage of several concepts to reduce the required input–output (IO) bandwidth, the necessary number of 156 chip pads organized in a pad ring still demanded a minimum die size of 14.3 mm^2 as seen in Fig. 8. The 100 K gate equivalents of the NeuroPipe-Chip covered about 70% of the core area.

The prototypes of the NeuroPipe-Chip have been tested. On a HP82000-tester we verified the correct logic functioning of the chip by applying functional test vectors from the benchmark simulation. We successfully tested the chip up to the maximum tester frequency of 95 MHz employing an external clock generator. Postlayout simulations showed a chip performance of 109 MHz. Extrapolating the measured power dissipation of the chip yields 2.5 W at 3.3 V and 100 MHz.

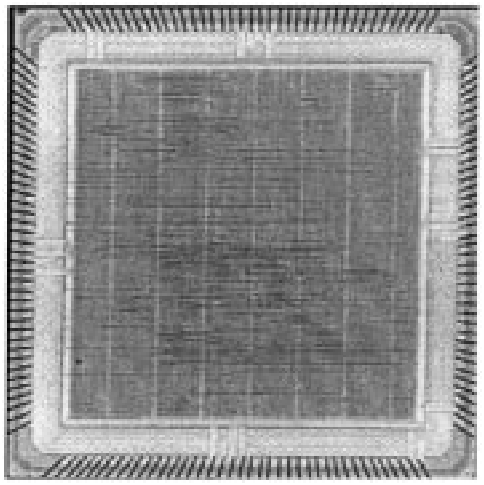
Fig. 8. Photograph of the NeuroPipe-Chip.

VIII. Conclusion

We presented a neuro-processor, the NeuroPipe-Chip, as part of an accelerator board concept which approaches real-time computational requirements for SANNs in the order of 10^6 neurons. Two new concepts were introduced on chip-level which lead to improved performance of the NeuroPipe-Chip. In the process of designing the NeuroPipe-Chip a VHDL-RTL-model of the chip was created. With an algorithmic description of the surrounding accelerator-system in behavioral VHDL, a system simulation was performed. For a simple SANN benchmark network for image segmentation, the simulation of the accelerator suggested about two orders of magnitude faster computation time than a 500 MHz Alpha workstation and a performance comparable to dedicated accelerator architecture consisting of 64 high-performance DSPs. The NeuroPipe-Chip comprising 100 K gate equivalents has been fabricated in an Alcatel five-metal layer 0.35 μm digital CMOS technology.

Acknowledgment

Prototypes of the Neuro-PipeChip have been fabricated and partly funded by the EUROPRACTICE service, funded by the IST program of the European Commission. The authors also gratefully acknowledge the support of U. Voss during chip testing.

References

[1] M. Arbib, "Background," in *Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pt. I, pp. 879–884.
[2] Cadence Design Systems Inc. *Silicon Ensemble Reference Manual, Vers. 5.2*, 1999.
[3] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, and H. J. Reitbock, "Coherent oscillations: A mechanism of feature linking in the visual cortex?," *Biol. Cybern.*, vol. 60, pp. 121–130, 1988.
[4] R. Eckhorn, H. J. Reitboeck, M. Arndt, and P. Dicke, "Feature linking via stimulus-evoked oscillations: Experimental results from cat visual cortex and functional implication from a network model," in *Proc. ICNN I*, 1989, pp. 723–720.
[5] G. Frank and G. Hartmann, "An artificial neural-network accelerator for pulse-coded model neurons," in *Proc. Int. Conf. Neural Networks ICNN95*, vol. 4, Perth, Australia, 1995, pp. 2014–2018.

[6] A. Jahnke, U. Roth, and H. Klar, "A SIMD/dataflow architecture for a neurocomputer for spike-processing neural networks (NESPINN)," in *Proc. MicroNeuro'96*, 1996, pp. 232–237.

[7] A. Jahnke, T. Schoenauer, U. Roth, K. Mohraz, and H. Klar, "Simulation of spiking neural networks on different hardware platforms," in *Proc. ICANN'97*. Berlin, Germany, 1997, pp. 1187–1192.

[8] A. Jahnke, U. Roth, and T. Schoenauer, "Digital simulation of spiking neural networks," in *Pulsed Neural Networks*, W. Maas and C. M. Bishop, Eds. Cambridge, MA: MIT Press, 1998.

[9] T. Lindblad and J. M. Kinser, *Image Processing using Pulse-Coupled Neural Networks*. Berlin, Germany: Springer-Verlag, 1998.

[10] C. v. d. Malsburg, "The correlation theory of brain function," in *Models of Neural Networks II*, Domany *et al.*, Eds. London, U.K.: Springer-Verlag, 1994, pp. 95–119.

[11] H. J. Reitboeck, M. Stoecker, and C. Hahn, "Object separation in dynamic neural networks," in *Proc. Int. Conf. Neural Networks ICNN93*, 1993, pp. 638–641.

[12] U. Schott and R. Eckhorn, "Internal Commun.," Philipps-Univ. Marburg, 1997.

[13] T. Schoenauer, N. Mehrtash, A. Jahnke, and H. Klar, "MASPINN: Novel concepts for a neuro-accelerator for spiking neural networks," in *Proc. VIDYNN'98*, 1998.

[14] Summit Design Inc. (1998) Visual HDL for VHDL—User's guide. [Online]. Available: http://www.summit-design.com

[15] Synopsys Inc.H. Klar, "Design compiler—Fundamentals, Ref. Manual, Vers. 98.08,", 1998.

[16] L. Weitzel, K. Kopecz, C. Spengler, R. Eckhorn, and H. J. Reitboeck, "Contour segmentation with recurrent neural networks of pulse-coding neurons," in *Sommer: Computer Analysis of Images and Patterns, CAIP Kiel*. Berlin, Germany: Springer-Verlag, 1997.

[17] G. Hartmann, G. Frank, M. Schafer, and C. Wolff, "SPIKE128K—An accelerator for dynamic simulation of large pulse-coded networks," in *Proc. MicroNeuro'97*, 1997, pp. 130–139.

[18] C. Wolff, G. Hartmann, and U. Ruckert, "ParSPIKE—A parallel DSP-accelerator for dynamic simulation of large spiking neural networks," in *Proc. MicroNeuro'99*, Granada, Spain, 1999, pp. 324–331.

[19] M. Ehlert and H. Klar, "Analog time-multiplexed hardware for spike-processing neural networks," in *Proc. MicroNeuro'97*, 1997, pp. 140–144.

[20] K. A. Boahen, "Communicating neuronal ensembles between neuromorphic chips," in *Neuromorphic Systems Engineering*, T. Lande, Ed. Boston, MA: Kluwer, 1998, ch. 11, pp. 229–261.

Tim Schoenauer received the Dipl.-Ing. degree in electrical engineering in 1995 from the Technical University of Berlin (TUB), Germany, after academic stays at the Ecole Nationale Superieure d'Electronique et de Radioelectricite de Grenoble (ENSERG), France, in 1994 and Stanford University, Stanford, CA, in 1995. He pursued the Ph.D. degree in electrical engineering at the TUB with research interests in the area of spiking neural networks and dedicated hardware for biology-oriented neural networks.

In 2000, he joined Multilink Technology Corporation, Munich, Germany, working on processors for high-speed high-bandwidth optical networks.

Sahin Atasoy received the Dipl.-Ing. degree in electrical engineering in May 2000 from the Technical University of Berlin, Germany.

In 2000, he joined Siemens AG, Berlin, Germany, working on the development of the basestations for third generation mobile radio networks (TDD).

Nasser Mehrtash received the Dipl.-Ing. degree in electrical engineering in 1996 from the Technical University of Berlin (TUB), Germany. He is currently pursuing the Ph.D. degree in electrical engineering at the TUB with research interests in the area of spiking neural networks.

Heinrich Klar (M'97) received the Dipl.-Ing. degree and the Dr.-Ing. degree from the Technical University of Munich, Germany, in 1972 and 1976, respectively.

In 1976, he joined the research Laboratories of Siemens AG, Munich, where he was engaged in the research development of circuits for transmission and processing of analog and digital signals as well as the design of standard MOS ICs. He now is a Professor at the Institute of Microelectronics of the Technical University of Berlin, Germany. His research interests are in integrated circuits for digital and analog signal processing, especially spiking neural networks, high-speed high-resolution analog-to-digital converters, and RF-CMOS circuits.