MASPINN: Novel Concepts for a Neuro-Accelerator for Spiking Neural Networks

T. Schoenauer, N. Mehrtash, A. Jahnke and H. Klar

Institute of Microelectronics, Technical University of Berlin
Jebensstr.1, Sekr. J13, D-10623 Berlin, Germany
Phone: +49 30 314-22640, Fax: +49 30 314-23029
E-mail: {tim, nasser}@mikro.ee.tu-berlin.de

ABSTRACT

We present the basic architecture of a Memory Optimized Accelerator for Spiking Neural Networks (MASPINN). The accelerator architecture exploits two novel concepts for an efficient computation of spiking neural networks: weight caching and a compressed memory organization. These concepts allow a further parallelization in processing and reduce bandwidth requirements on accelerator's components. Therefore, they pave the way to dedicated digital hardware for real-time computation of more complex networks of pulse-coded neurons in the order of 10^6neurons. The programmable neuron model which the accelerator is based on is described extensively. This shall encourage a discussion and suggestions on features which would be desirable to add to the current model.

Keywords: Spiking Neural Networks, Pulse-Coded Neural Networks, Neurochip, Neurocomputer, Neuro-Accelerator

1. INTRODUCTION

Biology is the source of inspiration for artificial neural networks. Spike processing (or simply: spiking) neuron models are closely related to biological neurons since they take into account the precise timing of spike events. This is not the case for conventional rate-coding neuron models as e. g. multilayer perceptrons, which are most commonly used for large networks. Spiking neural networks (also referred to as pulse-coupled neural networks) have gained increasing interest after experimental evidence of stimulus induced synchronized brain activity [Eck89, Gra89]. Synchronized firing of neuronal assemblies could serve the brain as a code for feature binding, pattern segmentation and figure/ground separation [Mal81, Eck88]. Since the human brain unlike technical systems easily solves such tasks as invariant object recognition, spiking neural networks (SNNs) are of great interest for machine vision applications. However, employing SNNs for image processing requires huge networks in the order of 10^6 of spiking neurons. Simulating large networks of complex neuron models is a computational expensive task and leads to high simulation times even for high-performance workstations. Furthermore, to solve real world tasks there is a need for real-time performance of complex networks, which can only be achieved by supercomputers or dedicated hardware.
At the University of Paderborn a neuroaccelerator for spiking neural networks, SPIKE128k, has been designed, implemented and tested [Har97]. However, the speed of this implementation is limited by the use of Field Programmable Gate Arrays (FPGA). A faster and more flexible implementation could be achieved by taking advantage of VLSI-technology using Application Specific Integrated Circuits (ASICs). Such an approach was pursued by the NESPINN-System (Neurocomputer for Spike-Processing Neural Networks) [Jah96]. Since image processing applications with spiking neurons like in [Wei97] have shown an even higher demand for network complexity in order to allow a sufficient resolution of processed images [Sch97], concepts for a second generation NESPINN-system have been developed: MASPINN (Memory Optimized Accelerator for Spiking Neural Networks). In comparison to the NESPINN-Architecture, the memory organization and dataflow of MASPINN is optimized to tackle the challenge of simulating more complex networks.
Before describing the concept of MASPINN, spiking neuron models and the derivation of a generic programmable neuron

model as well as typical network characteristics are outlined. Subsequently, the major steps simulating such networks are reviewed. Concepts how to increase the efficiency of simulation of SNN are discussed. Two new concepts of accelerator design are introduced. It is shown how these concepts are realized in the neuroaccelerator architecture MASPINN. Finally the performance of MASPINN is discussed.

2. Spiking Neural Networks

2.1 Neuron Models

Spiking neuron models are biophysical models which try to account for properties of real neurons without descending to the level of ionic current but by modelling the integrated signal flow through parts of the neuron, in particular the synapses, axon, dendrites and soma. Spiking neurons represent complex leaky Integrate-and-Fire neurons. The rudimentary behavior of spiking neurons is the following: Input spikes from presynaptic neurons are weighted and summed up yielding a value called membrane potential. The membrane potential is time dependent and decays when no spikes are received by the neuron. If however spikes excite the membrane potential sufficiently so that it exceeds a certain threshold, a spike is emitted. After the emission of a spike the neuron is unable to spike again for a certain period called refractory period. A generic pulse-coded neuron model, referred to as Spike Response Model (SRM), can be described mathematically by [Ger98]:

(1) $$u_i(t) = \sum_{t_i^{(f)} \in F_i} \eta_i(t - t_i^{(f)}) + \sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in F_j} \varepsilon_{ij}(t - t_j^{(f)})$$

(2) $$F_i = \{t_i^{(f)}; (1 \le f \le n)\} = \{t \mid u_i(t) = \upsilon\}$$

(3) $$\Gamma_i = \{j \mid j \text{ presynaptic to } i\}$$

where	$t_i^{(f)}$, $t_j^{(f)}$	firing time of neuron *i* and neuron *j*, respectively.
	$u_i(t)$	membrane potential of neuron *i*.
	F_i , F_j	set of all firing times of neuron *i* and neuron *j*, respectively.
	υ	if threshold υ is reached by $u_i(t)$, neuron *i* emits a spike.
	Γ_i	set of neurons *j* presynaptic to *i*.
	$\eta_i(\cdot)$	kernel to model the refractory period.
	$\varepsilon_{ij}(\cdot)$	kernel to model the postsynaptic potential of neuron *i* induced by a spike of neuron *j*.

Eq. (1) yields the membrane potential $u_i(t)$ of neuron *i* and contains in the second addend the sum of all spikes from presynaptic neurons *j*. Spikes from presynaptic neurons are evaluated by a kernel function $\varepsilon_{ij}(\cdot)$ which models the response to presynaptic spikes. When the membrane potential $u_i(t)$ exceeds the threshold υ, neuron *i* spikes and inhibits itself for a refractory period. This is modeled by a negative feed back which the first addend of Eq. (1) accounts for by summing all spikes emitted by neuron *i* at $t_i^{(f)}$ and evaluating these spikes with a kernel function $\eta_i(\cdot)$. Typical kernel functions $\eta_i(\cdot)$ and $\varepsilon_{ij}(\cdot)$ are:

(4) $$\eta_i(s) = -\eta_0 \cdot e^{-s/\tau} \cdot H(s) \qquad \varepsilon_{ij} = w_{ij}\frac{1}{1-(\tau_s/\tau_m)}[e^{-s/\tau_m} - e^{-s/\tau_s}] \cdot H(s)$$

where τ, τ_m and τ_s are time constants and H(s) denotes the heavyside stepfunction. The SRM described by Eq. (1)-Eq. (3) can be extended in order to model a synchronization of neuronal assemblies by introducing linking connections [Eck89]. Inputs from the linking connections modulate the feeding inputs, which correspond to the second addend in Eq. (1). The integrated signals from the linking inputs, together with a constant offset term, (+1), interact multiplicatively with the integrated signals from the feeding inputs. Eq. (1) becomes in this case

(5) $$u_i(t) = \sum_{t_i^{(f)} \in F_i} \eta_i(t - t_i^{(f)}) + \left[\sum_{j \in \Gamma_{fi}} \sum_{t_{fj}^{(f)} \in F_{fj}} \varepsilon_{fij}(t - t_{fj}^{(f)})\right] \cdot \left[1 + \sum_{j \in \Gamma_{li}} \sum_{t_{lj}^{(f)} \in F_{lj}} \varepsilon_{lij}(t - t_{lj}^{(f)})\right]$$

where the extra index *f* in Γ_{fi}, $t_{fj}^{(f)}$, F_{fi}, ε_{fij} marks values related to feeding connections, while the extra index *l* in Γ_{li}, $t_{lj}^{(f)}$, F_{li}, ε_{lij} marks values related to linking connections. Such a model shall be called extended Spike Response Model. A graphical model of the extended SRM is shown in Fig. 1.

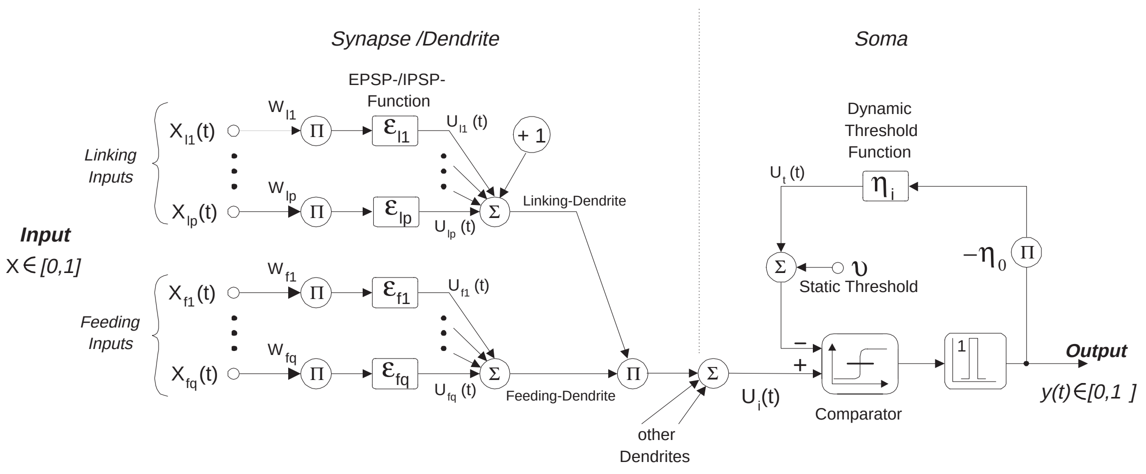

Figure 1: Graphical representation of the extended Spike Response Model.

Digital hardware requires a model discrete in time. In Fig. 2 simple leaky accumulators have been chosen as IPSP-/EPSP-filter functions $\varepsilon_{ij}(\cdot)$ and the refractory period filter function $\eta_i(\cdot)$. Also in Fig. 2 delay elements at the output of the neuron are added to model axonal delays. As one can see in Fig. 2, in order to memorize the state of a neuron the filter values $u_t(nT)$, $u_{fx}(nT)$ and $u_{lx}(nT)$ need to be stored. These values will be further on called *internal potentials* or *IP-values*.

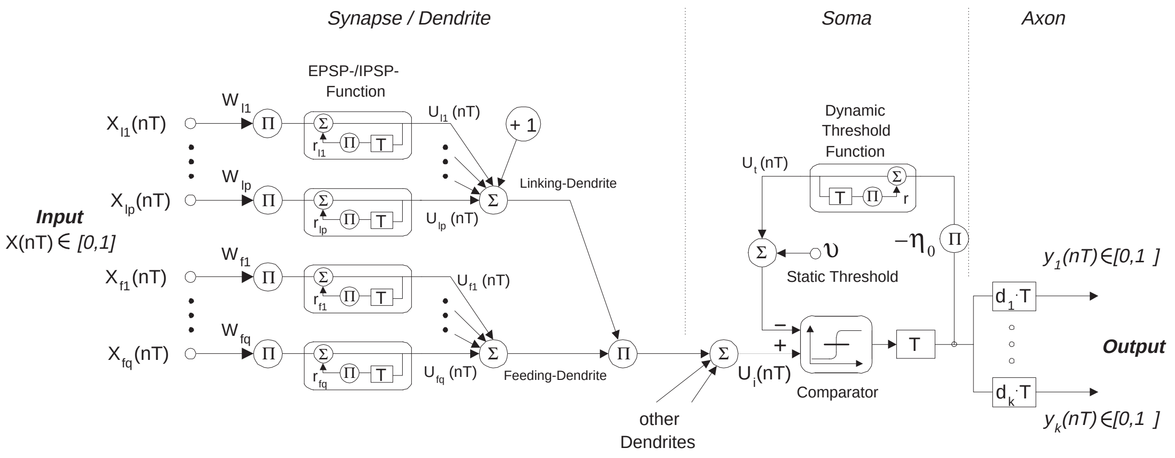

Figure 2: Time-discrete extended Spike Response Model with first-order filters.

The extended Spike Response Model as described by Eq. (5) covers to our knowledge all features inherent to spiking neuron models which are subject of active research. To allow for different types of neurons of the extended SRM, a configurable (so called programmable) model neuron is desirable for a hardware implementation. It will be presented in subsection 3.3.

2.2 Network Topology and Characteristics

As SNNs we assume sparsely connected networks. The network is composed of several layers where each layer consists of neurons with equal properties. Neurons may be connected laterally (within one layer) or to other layers. Connections are usually structured in perceptive fields, but in a neuroaccelerator system it should be possible to choose them arbitrarily. The network activity resembles the average number of spikes per time slice divided by the total number of neurons. Network activity is typically low e.g. below 1% for SNNs.

3. Computing Spiking Neural Networks

3.1 Basic Procedures

In a digital simulation, a SNN is computed at discrete times $n \cdot T$, where the period T is called time slice. During each time slice, the next state of the network is computed. Let us assume the SNN consists of spiking neurons as shown in Fig. 2.

Four major steps need to be taken:

1. **Decay of IPs:**
   Internal potentials like $u_t(nT)$, $u_{fx}(nT)$, $u_{lx}(nT)$ are decreased according to kernel functions $\eta_i(\cdot)$ or $\varepsilon_{ij}(\cdot)$. In case of a simple leaky accumulator this would correspond to a multiplication with a decay factor r_x (see Fig. 2).
2. **Spike propagation:**
   Spikes are propagated from the outputs of neurons (source-neurons) that e.g. spiked during the previous time slice to the inputs of connected neurons (target neurons).
3. **Output:**
   The output function combines all IP-values of a neuron to the membrane potential $u_i(nT)$ and a spike is emitted or not depending on the comparison with a threshold.
4. **Learning:**
   Learning parameters are adapted according to a learning algorithm.

The concepts discussed in this paper focus on step 1-3.

3.2 Concepts of Efficient Simulation

To efficiently simulate SNNs several strategies can be pursued: event list protocol, sender-oriented connection list, fixed-point arithmetic and neglecting small IP-values [Jah98].

Event-list: To implement step 2 (Spike Propagation), all spikes of the previous time slice[1] need to be distributed according to the connections of source neurons[2] to their target neurons. An event list protocol is an efficient communication scheme [Laz93]. The event list contains the addresses of source neurons. Due to a low network activity, only a small fraction of the total number of neurons needs to be stored in that list.

Sender-oriented connection list: During step 2 all connections from these source neurons as well as the corresponding weights need to be determined. It is suitable to store connections and weight values in a list. Such a connection list contains either for each source neuron address all target neuron addresses (sender-oriented) or for each target neuron address all source neuron addresses (receiver-oriented) [Fra95]. A sender-oriented connection list is appropriate since it ensures that only active connections are computed.

Neglecting small internal potentials: Since the network activity is low and the IP-values decay in time, many of the IP-values approach or already are zero. The percentage of negligible IP-values depends on network parameters and stimuli. For an enlarged version of a simple SNN presented by Reitboeck et al. [Rei93], we found an average number of negligible IP-values in the order of 80%. It would be a waste of computational resources to apply a filter function to these values. Rather each IP-value could be tagged as valid (tag=1) or not (tag=0) after a comparison with a minimum value IP_{min}. In this case, only the IP-values with a valid tag-bit are processed.

Numeric precision: Instead of using floating-point arithmetic, fast fixed-point arithmetic may be used without any degradation in network performance. According to the resolution analysis of Roth et al. [Rot95], accuracies from 8bit (weights) up to 16bit (internal potentials) are sufficient for vision problems.

3.3 A Programmable Model of a Spiking Neuron

A programmable neuron model admits to configure spiking neurons of various complexity and with a different parameter set. Such a model is shown in Fig. 3. It represents a configurable neuron which may be programmed to correspond to the neuron model in Fig. 2. Furthermore, the programmable neuron allows several inputs $x(nT)$ to one IP-filter in order to reduce computational expense by letting spikes that can be modelled by the same filter characteristic split one filter. Otherwise it consists of summing nodes to sum weighted spikes which share a filter, a processing unit which combines the IP-values according to a program code, a comparator which compares the resulting value with e.g. an activity threshold and delay elements at the output. The parameter set of the neuron comprises e.g. the filter coefficients, program code, threshold values and delay values. The program code specifies how the IP-values are combined: if they are e.g. added, subtracted, multiplied or cascaded. By cascading the IP-filters, higher-order filters can be configured e.g. to emulate a kernel function with a rise and a subsequent decay as denoted by the right part of Eq. (4). In Fig. 3, cascading of IP_0 and IP_1 is depicted with a broken line, which can be specified by the program code for IP_0.

Events in neural networks occur at a time scale of milliseconds and modern VLSI-circuits operate at a time scale of about ten nanoseconds. Therefore, within the time scale of the functioning of a biological neuron, one VLSI-implementation is able to simulate the function of many neurons. Taking advantage of the higher speed of VLSI-circuits, parallelization may be limited to only a few processing units while still achieving real-time simulations of 10^5 neurons and more. However the values which are not currently calculated must be stored in memory. This is the reason why e.g. employing the neuron model in Fig. 3

1. This simplification is valid in case none of these spikes has an extra-delay.
2. Source neurons are here referred to as neurons emitting a spike while target neurons receive a spike.

requires memories to store: the IP-values and the parameter set. If small IP-values are to be neglected, a tag memory is required, too.

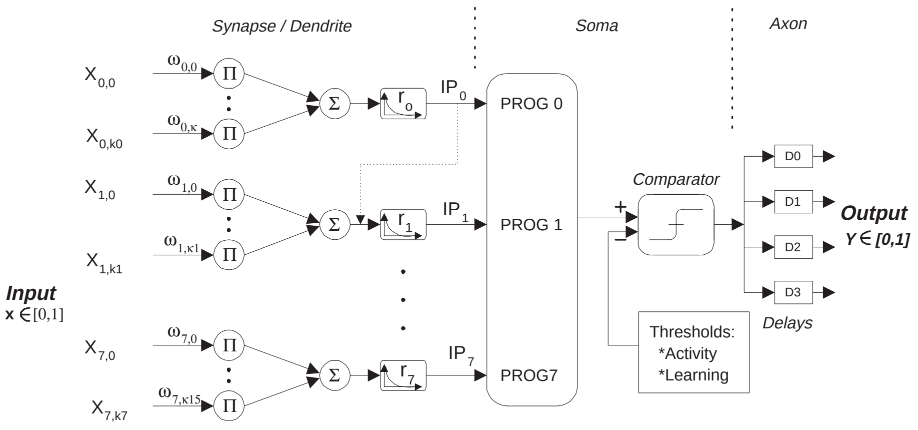

Figure 3: A generic programmable model of a spiking neuron

4. Concepts of the MASPINN-Architecture

4.1 Simulation of Complex Spiking Neural Networks

When designing SNN-accelerators for real-time simulation of more complex SNNs, two things need to be considered at first: memory capacity has to increase to store more network variables and processing speed must rise to process more network variables within the same time frame. Due to the continuous growth of memory capacity during the past years, capacity of off-chip memory is a minor problem. However, the bandwidth of data-transfer between a memory chip and a processing chip has not grown correspondingly. It is limited by the number of pins supplied by the chip package and its operation frequency. Both have only slightly increased. Therefore, bandwidth may become a limiting factor. System speed may be formulated as

$$v_{sys} = f_{clock} \cdot N_{PE} \cdot \xi_{PE} \tag{6}$$

with a circuit operation frequency f_{clock}, N_{PE} processing elements working in parallel and the efficiency of these processing elements ξ_{PE} defined as the reciprocal value of the number of clock cycle it takes one processing element to process one IP-value. Clock frequency rises with the downscaling of VLSI-technology and must be presumed constant for a certain state of technology. Suppose the hardware effort (e.g. chip size, pin count, board complexity) is to be kept constant also, there are two ways left to increase the system speed: simplifying processing elements so that a higher number N_{PE} may be chosen without increasing the overall hardware effort or increasing the efficiency of the processing elements without increasing their complexity. We will concentrate on the latter strategy. Therefore, the task of designing an accelerator for more complex SNNs is conform with the optimization of bandwidth requirements and the efficiency of the processing elements.

4.2 Limitations of Accelerator Architectures

In subsection 3.2, concepts of efficient simulation as a spike event list, a sender-oriented connection list and neglecting small IP-values have been presented. Taking advantage of a spike event list already suggest a basic architecture of a system as shown in Fig. 4a. The major system elements are a Neuron Unit, a Connection Unit and a Spike Event List. The Neuron Unit performs the computation of each neuron in the network (e.g. as suggested by the programmable model in Fig. 3). Besides continuously updating the IPs, the Neuron Unit supplies the addresses of neurons which emit a spike: the source neurons. They will be stored in the Spike Event List. In the following time slice, the Connection Unit reads the addresses of the relevant source neurons and supplies the addresses of the target neurons with the corresponding weights to the Neuron Unit. This basic organization has been used by NESPINN as well as SPIKE128k.

It requires a sequential processing of steps 1-3 decay, propagation and output (see subsection 3.2). In principle the decay of IP-values and spike propagation can be performed within the same time period. However each step requires a separate memory access: while the decay of potentials can be organized as a regular dataflow, propagating spikes is irregular since the occurrence of spikes is not predictable. Since both steps require an access to the same memory, they cannot be performed in parallel. We suggest a new organization of the system which allows the processing of all three steps in parallel. Its fundamental organization as shown in Fig. 4b is quite similar to the one in Fig. 4a, however it is based on two new concepts:

weight caching and a compressed IP-memory.

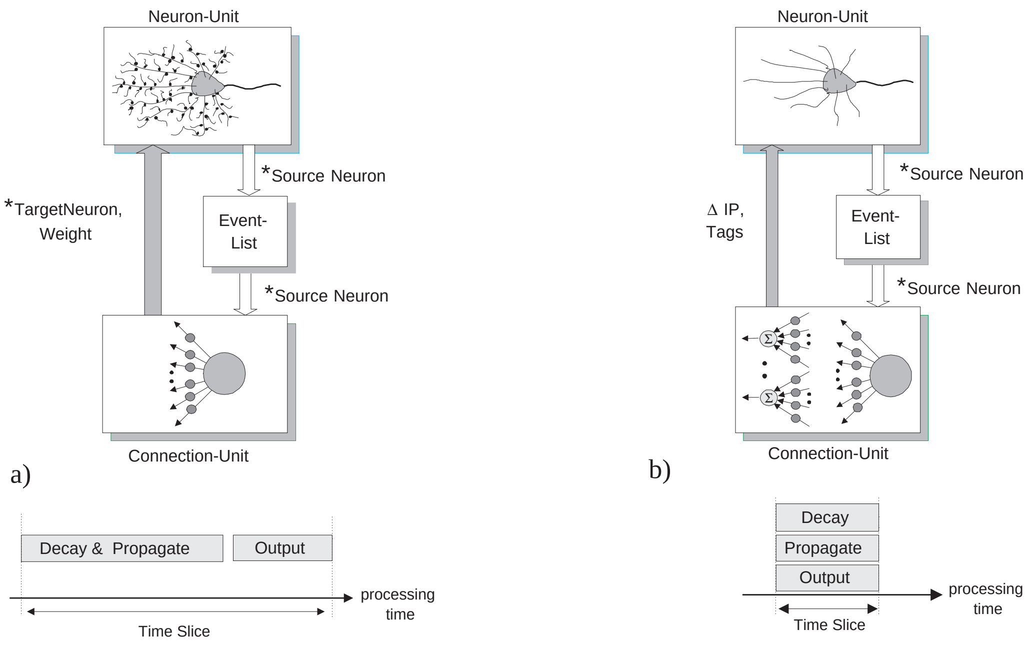

Figure 4: Basic organization of an accelerator system employing a spike event list (' * ' denotes 'address of'). a) typical b) MASPINN

4.3 Novel Concepts: Weight Caching and Compressed IP-Memory

Weight Caching: instead of sending each weight and the corresponding target neuron address separately from the Connection Unit to the Neuron Unit, they are accumulated to values ΔIP in a copy of the IP-memory (weight cache) in the Connection Unit. During the next time step the accumulated weights are sent to the Neuron Unit which then performs the three steps: decay of potentials, propagating of (accumulated) spikes and computing the output. Since in this case the dataflow is regular, this processing can be fully pipelined.

Compressed IP-Memory with tag-bit addressing: it was pointed out earlier, that due to low network activity and decaying characteristics of IP-values many IP-values are negligible. As illustrated in Fig. 5, instead of saving a memory location for each IP-value - no matter if it is relevant or not - the compressed IP-Memory stores only the non-negligible IP-values one after the other. This optimization in memory organization combined with tag-bit addressing scheme has several advantages such as a smaller memory size and a more efficient memory access, which is crucial for the simulation of even more complex networks.

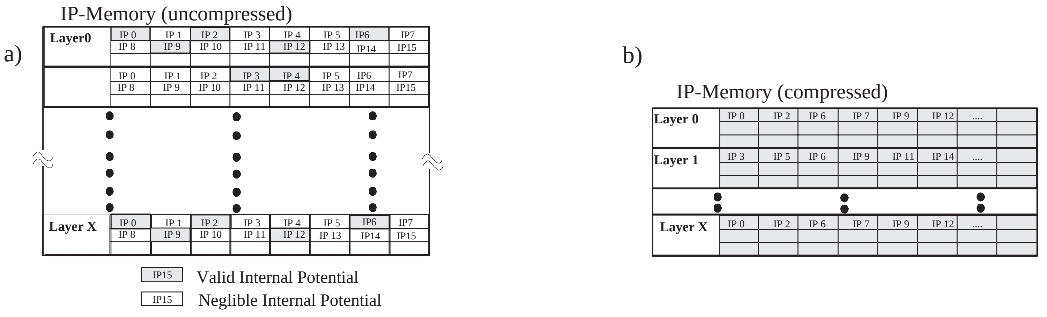
IP-Memory (uncompressed)
a)
Layer0
Layer X
IP 0 IP 8 IP 1 IP 9 IP 2 IP 10 IP 3 IP 11 IP 4 IP 12 IP 5 IP 13 IP6 IP14 IP7 IP15
IP15 Valid Internal Potential
IP15 Neglible Internal Potential
b)
IP-Memory (compressed)
Layer 0
IP 0 IP 2 IP 6 IP 7 IP 9 IP 12 ....
Layer 1
IP 3 IP 5 IP 6 IP 9 IP 11 IP 14 ....
Layer X
IP 0 IP 2 IP 6 IP 7 IP 9 IP 12 ....

Figure 5: Internal Potentials are stored in the IP-Memory. In a) each IP-value holds a fixed memory location while in b) only relevant IP-values are stored one after each other.

4.4 Neuron-Unit

The Neuron Unit may perform the three steps decay, propagate and output fully pipelined and with several processing units in parallel. Pipelining is possible since the dataflow is regular. The data processed during one time slice (IP, ΔIP, tags) is completely available at the beginning of the time slice. This is the case, because the Connection Unit has already accumulated the weighted spikes (ΔIP) during the previous time slice. Now, the Neuron Unit simply needs to add these accumulated weights ΔIP to the decayed IP-values. This can be done consecutively e.g. starting with the first IP-value of the first layer until the last IP-value of the last layer. In doing so, each IP-value is processed pipelined where at the end the IP-values are combined to the membrane potential according to the program code and if it exceeds the dynamic threshold the address of the corresponding neuron (source neuron) will be sent to the Spike Event List.

As illustrated in Fig. 6, the Neuron Unit consists of a Neuron Chip responsible for all the processing and memories to store the IP-values, the IP-tags and the network parameters. Network parameters contain the parameter set of the programmable neuron (see subsection 3.3). Since a layer consists of neurons with equal properties, the network parameter memory has to be accessed only once per layer. On the other hand the IP-memory and the IP-tag-memory need to be accessed constantly. To achieve a high system speed parallel processing is desirable. That means several IP-values and IP-tag values have to be accessed in a one clock cycle as well as ΔIPs need to be read from the Connection Unit. Hence, there is a demand for a high IO-bandwidth of the Neuron Chip. Not only the values themselves but also the addresses of IP-values and ΔIPs contribute to the requirement of a high IO-bandwidth. For example a complex network of 10^5 neurons with 6 IP-values each, the IP-address would have a bitwidth of 20bit. Soon the required IO-bandwidth exceeds the bandwidth achievable by ASICs, since the bandwidth of ASICs is bound by a limited number of pins.

The use of the embedded DRAM-technology would be an attractive alternative. On-chip memory exhibits a virtually unlimited bandwidth. The compressed IP-memory concept is very attractive for embedded-DRAM design, as up to 75% of memory capacity could be saved. Since area is a major concern in this extremely expensive technology, a less expensive design or a design for the computation of far more complex networks could be fabricated. Unfortunately, the costs of the embedded-DRAM technology is currently prohibitively high for small volume designs.

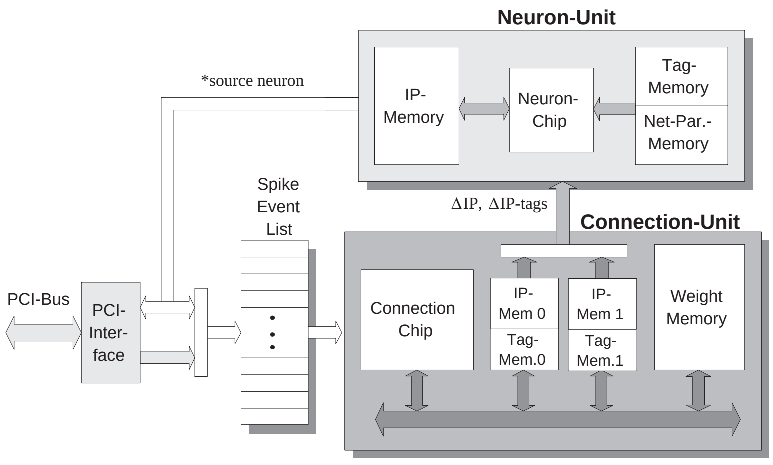

Figure 6: Internal Structure of Connection Unit and Neuron Unit employing the weight caching and compressed IP-Memory concept.

But a compressed organization of the IP-memory exhibits also advantages for off-chip memories, where memory capacity is not of concern. Instead of reading IP-values at different addresses in different memory devices, one wide word consisting of the next relevant IP-values can be easily accessed from one memory device by simply incrementing the address pointer. Therefore a compressed IP-memory facilitates memory access.

However storing only some of the IP-values also causes a problem: the assignment of a certain IP-value to a certain memory location is lost (see Fig. 5). How can the information be obtained which IP-value is stored in what location of the memory? To store the address of each IP-value in memory is not a solution, since for each IP-value an extra value would need to be accessed and also memory resources would be massively wasted. A very elegant way is the extraction of the address from the IP-tag bits. Since the tag bits contain the information which IP-value is valid and which is not valid, the address of all valid IP-values can be decode from the tag bits if IP-values are accessed consecutively. Assuming an average validity of IP-values of 20% would mean that in average only 5bits per IP-value need to be transferred to obtain the IP-address. This is a small value considering that a complex network of 10^5 neurons with 6 IP-values each requires a bitwidth of 20bit for the absolute IP-address. Since tag bits are handled by the Neuron Unit anyway to prevent the computation with irrelevant data, no extra resources are required for this kind of address encoding. The Neuron Unit - as described in this subsection - exhibits a high efficiency ξ_{PE} of its fully pipelined processing elements due to the concept of weight caching. Furthermore the concept of a compressed IP-Memory in conjunction with a tag-bit addressing scheme reduces the IO-bandwidth requirements on the Neuron Chip.

4.5 Connection-Unit

As illustrated in Fig. 4b, the Connection Unit receives source neuron addresses and supplies ΔIPs (accumulated weights) and ΔIP-tags. ΔIP-tag bits label the corresponding ΔIP-values as valid (ΔIP-tag='1') or not valid (ΔIP-tag='0'). In Fig. 6, the internal structure of the Connection Unit is depicted. Major components of the Connection Unit are the weight memory, two copies of the complete (not-compressed) IP-Memory and a Connection Chip. The two copies of the IP-Memory -in the following called weight caches- serve the accumulation of weights during a time-slice. They have two functional roles.

One weight cache is receiver. From the connection list it receives weights and a corresponding target address. The target address corresponds to a certain memory location in the weight cache. If there is already a relevant value stored in that location (ΔIP-tag='1'), the weight needs to be added to that value. Otherwise the weight is simply written to that location and ΔIP is set to '1'. Thereby all corresponding weights are accumulated at the appropriate location in the weight cache.

The other weight cache functions as a sender. According to the ΔIP-tag, it reads out the relevant ΔIP-values and sends them to the Neuron Unit. Weight caches switch roles each time slice: one time-slice they preprocess the data which they received from the connection list, the next time slice they supply this data to the Neuron Unit. Meanwhile the other weight cache performs exactly the opposite operation.

Advantage of this concept is that the ΔIP-values are generated one time-slice ahead, and therefore allow the parallelization of the three steps of decay, propagation and output. Also a major benefit is that the IO-bandwidth requirements for the Neuron Chip is further reduced: instead of sending each weight and target neuron address separately (see Fig. 4b), only one accumulated ΔIP-value needs to be transferred. If for example one neuron receives ten spikes during one time slice, instead of sending each weight with a corresponding target address separately ten times, it sends only one ΔIP-value and some tag-bits. Further reduction of the bandwidth requirements on the Neuron Chip is achieved by employing the efficient tag-bit addressing scheme[1].

Obviously these advantages have to be payed with an extra complexity of the Connection Unit mainly by the weight caches and its peripheral logic. Fast SRAMs may be used as weight caches and the logic function that needs to be implemented is a simple read-accumulate-write action. However, for very high connectivity, the performance of the weight caches might become the bottle neck of the system and need to be designed carefully.

5. Performance Evaluation

The concept of weight caching and compressing the IP-memory lead to a speed-up compared to the system organization in Fig. 4a since the three steps[2] to compute an SNN may be performed in parallel. Ideally, there would be a speed-up factor of three. However, this assumes that all three steps require the same processing time. In reality this is not the case and the step with the longest processing time will determine the duration of the computation of a time slice. However, it is reasonable to assume the same order of magnitude for processing time of the three steps. The system organization allows a fully pipelined processing of the neuron function (Neuron Chip). Hence, in the ideal case each processing element has a throughput of one IP-value (performing decay, propagation and output) per clock cycle. Assuming a MASPINN-Architecture employing a Neuron Chip with four processing elements, a performance estimation for the previously mentioned network [Rei93] yields the results stated in Tab.1 for a network complexity of up to 10^6 neurons with three IP-values each.

Number of neurons	Number of IPs	Ultra-Sparc 300MHz 1 PE	Pentium II 266MHz 1 PE	Alpha 500MHz 1 PE	CNAPS 50MHz 256PE	CM-2 10MHz 16KPE	NESPINN 50MHz 4PE	MASPINN 100MHz 4PE
16K	48K	11ms	10ms	9ms	1.5ms	<0.01ms*	0.38ms*	0.06ms*
128K	384K	84ms	85ms	67ms	~1s*	<0.1ms*	(3.0ms)*	0.39ms*
512K	1,5M	380ms	427ms	291ms	>>1s*	~1ms*	(11.7ms)*	1.56ms*
1M	3M	730ms	not avail.	650ms	-	not avail.	(23.4ms)*	3.1ms*

Table 1: Performance of the MASPINN-architecture compared to various other hardware platforms (real-time requirement: time slice < 1ms). Values marked by * are estimated, other values are measured (Assumption: max. 0.5% network activity, 20% valid IPs).

1. The efficiency of the tag-bit addressing scheme depends of course on the number of ΔIP which are relevant. If a very small number only is relevant, the tag-bit scheme becomes inefficient. However, it would not matter, since in such a case bandwidth requirements would be extremely small anyway.
2. Learning is not considered.

6. Conclusion

The neuron model which the MASPINN-system is based on was discussed. Typical characteristics of spiking neural networks were outlined and basic concepts of neuroaccelerator design were pointed out. Two novel concepts for an accelerator architecture for spiking neural networks have been proposed: weight caching and a compressed internal potential memory. At the cost of additional memory units - the weight caches - the concepts of weight caching allow a further parallelization when computing spiking neural networks and therefore leads to a speed-up. Furthermore, weight caching admits a regular dataflow opposed to a naturally irregular dataflow due to unpredictable spike events in previous systems. The regular dataflow facilitates the computation and a fully pipelined processing is feasible. The concept of a compressed internal potential memory simplifies memory access. It is also a very attractive concept for designs where memory is very expensive (e.g. on-chip memory, embedded DRAM), since the required capacity of memory storing the internal potentials can be reduced up to 80%. On-chip memory is attractive for accelerators for spiking neural networks as their computation is IO-bounded. Weight caching and a compressed internal potential memory as well as previously applied concepts like an event list protocol, a sender-oriented connection list, fixed-point arithmetic and neglecting irrelevant internal potentials constitute the backbone of the proposed <u>M</u>emory Optimized <u>A</u>ccelerator for <u>Spi</u>king <u>N</u>eural <u>N</u>etworks (MASPINN). The estimated performance is compared to the performance of a variety of other hardware platforms and suggests that MASPINN may perform real-time simulation of up to 10^6 simple neurons with four fully pipelined processing elements. Currently, a detailed specification of the MASPINN-system is elaborated.

7. Acknowledgments

We would like to thank Ulrich Schott, Ulrich Jagdhold and Thomas Geiger for providing simulation results of benchmarkfiles.

8. References

[Eck88] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, H.J. Reitböck, "Coherent oscillations: A mechanism of feature linking in the visual cortex?", Biological Cybernetics 60:121-130, 1988.

[Eck89] R. Eckhorn, H. J. Reitboeck, M. Arndt, P. Dicke, "Feature linking via stimulus-evoked oscillations: Experimental results from cat visual cortex and functional implication from a network model", *Proc. ICNN* I: 723-730, 1989.

[Fra95] G. Frank, G. Hartmann, "An artificial neural network accelerator for pulse-coded model-neurons", Proceedings on International Conference on Neural Networks (ICNN), Perth, 4:2014-2018, 1995.

[Ger98] W. Gerstner, "Spiking Neurons", In: *Pulsed Neural Networks*, W. Maas and C.M. Bishop (Eds.), MIT Press, 1998.

[Gra89] C. M. Gray, W. Singer, "Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex", *Proc. Natl. Acad. Sci. USA* 86: 1698-1702, 1989.

[Har97] G. Hartmann, G.Frank, M.Schaefer, C.Wolff, "SPIKE128K - An accelerator for dynamic simulation of large pulse-coded networks", *Proceedings of the 6th International Conference on Microelectronics for Neural Networks, Evolutionary & Fuzzy Systems*, Dresden, pages 130-139, 1997.

[Jah96] A. Jahnke, U. Roth, H. Klar, "A SIMD/Dataflow Architecture for a Neurocomputer for Spike-Processing Neural Networks (NESPINN)", *MicroNeuro'96*, pages 232-237, 1996.

[Jah98] A. Jahnke, U. Roth, T. Schoenauer, "Digital Simulation of Spiking Neural Networks", In: *Pulsed Neural Networks*, W. Maas and C.M. Bishop (Eds.), MIT Press, 1998.

[Laz93] J. Lazarro, J. Wawrzynek, "Silicon Auditory Processors as Computer Peripherals", Advances in Neural Information Processing Systems 5: 820-827, 1993.

[Mal81] C.v.d. Malsburg, "The correlation theory of brain function", Internal Report 81-2, MPI für Biophysikalische Chemie, Göttingen, 1981. Reprinted in *Models of Neural Networks II*, Domany et al. (Eds.), Springer, pages 95-119, 1994.

[Sch97] U. Schott, R. Eckhorn (Philips-Universität Marburg), internal communication, 1997.

[Wei97] L.Weitzel, K.Kopecz, C.Spengler, R.Eckhorn, H.J.Reitboeck, "Contour segmentation with recurrent neural networks of pulse-coding neurons", In: Sommer, Daniilidis, Pauli: Computer Analysis of Images and Patterns, CAIP Kiel, Springer Verlag, 1997.