Faculty of Physics and Astronomy

University of Heidelberg

Diploma thesis

in Physics

submitted by

Bernhard Kaplan

born in Heidelberg

December 2008

Self-Organization Experiments for a Neuromorphic Hardware Device

This diploma thesis has been carried out by Bernhard Kaplan at the

Kirchhoff Institute for Physics

University of Heidelberg

under the supervision of

Prof. Dr. Karlheinz Meier

Abstract

Self-Organization Experiments for a Neuromorphic Hardware Device

This thesis presents methods, investigations and experiments aiming at the realization of self-organization experiments on a neuromorphic hardware device. The utilized chip implements a network of leaky integrate-and-fire neurons and spike-timing dependent plasticity in every synapse, operatable with a high speedup factor compared to biological real-time. In order to exploit these features, a self-optimizing winner-take-all architecture is chosen which represents a task with a widely adjustable complexity. Preparatory studies are presented and methods introduced which solve problems on the way towards the winner-take-all implementation in hardware, often utilizing the software simulator NEST. A purely spike-based technique for testing the total conductance of a silicon membrane is proposed and successfully applied in both hardware and NEST. A quality measure for cross-inhibition architectures is defined and utilized for extensive studies on parameters for an appropriate cross-inhibition operation regime in hardware. The self-optimization of a winner-take-all classificator via long-term synaptic plasticity is set up and investigated using the NEST simulator. Remaining technical obstacles which avoid the application of this experiment in hardware are analysed. Most of them are chip inherent and will be addressed by future revisions of the hardware.

Selbstorganisationsexperimente für eine Neuromorphe Hardware

Die vorliegende Arbeit präsentiert Methoden, Untersuchungen und Experimente zur Realisierung von Selbstorganisationsexperimenten auf einem neuromorphen Hardwaresystemssystem. Der verwendete Mikrochip implementiert ein Netzwerk von leaky integrate-and-fire Neuronen mit Aktionspotential-Intervall-abhängiger Plastizität (STDP) in allen Synapsen. Relativ zur biologischen Echtzeit läuft das System mit stark erhöhter Geschwindigkeit. Um diese Eigenschaften nutzbringend anzuwenden, wird eine selbstoptimierende Winner-Take-All Architektur ausgewählt, deren Komplexität in der Problemstellung über einen großen Bereich skaliert werden kann. Vorbereitende Studien und Methoden zur Lösung von Schwierigkeiten bei der Implementierung dieser Architektur auf der Hardware werden vorgestellt. Hierzu wird häufig der Simulator NEST verwendet. Eine spikebasierte Methode zum Testen der Gesamtkonduktanz einer künstlichen Membran wird vorgestellt, die sowohl auf der Hardware als auch im Simulator NEST verwendet wird. Verbleibende technische Hürden, die die Verwendung dieser Architektur in Hardware vereiteln, werden untersucht. Die meisten sind auf Chipeigenschaften zurück zu führen und werden in späteren Revisionen der Hardware berücksichtigt werden.

Contents

Introduction

The human brain provides us with many impressive abilities we take for granted, but we still do not understand many of its underlying concepts. In nearly every situation, the brain has to cope with large amounts of data and can extract the important features of its environment within a fraction of a second, e.g. when recognizing faces. Furthermore, most tasks are solved with a high adaptability regarding changes in the environment on different time scales, which makes the brain an extremely flexible computing 'device'. In contrast to psychological or philosophical approaches, efforts in neuroscience - and even more in computational neuroscience - mainly focus on this information processing aspect in order to understand how the brain works.

One of the most important features to be considered in this attempt is the brain inherent plasticity, which leads to permanent changes in the momentary activity and in the network structure by short-term [*Tsodyks and Markram*, 1997] and long-term plasticity [*Levy and Steward*, 1983; *Bi and Poo*, 1997; *Dan and Poo*, 2004] effects. Structural plasticity is the most promising candidate to explain learning and memory formation physiologically [*Neves et al.*, 2008]. To learn and remember over different time scales and to apply the recovered information in various contexts is what many people would call an essential component of intelligent behavior.

Comprehensive biophysical [*Holmes and Levy*, 1990] as well as phenomenological models [*Song et al.*, 2000; *Worgotter and Porr*, 2005; *Pfister et al.*, 2006] are still not able to describe and predict the complex and interweaved phenomena of life-spanning brain development, formation of various kinds of memory, adaptation to changing environments and many other forms of neuronal self-organization. First theoretical analyses of learning on the neuron level were done by Donald Hebb in 1949 [*Hebb*, 1949], which led to the development of so-called rate-based learning rules [*Bienenstock et al.*, 1988; *Dayan and Abott*, 2001]. For a long time, rate-based codes were assumed to be the key coding mechanism in neural information processing. Since fundamental experimental evidence was found which indicated the influence of the precise temporal structure of inputs on the resulting structural modification in the brain [*Levy and Steward*, 1983; *Bi and Poo*, 1997], temporal codes play an increasingly important role in models of learning and information processing [*Izhikevich*, 2005; *Morrison et al.*, 2008].

Using temporal codes requires a high detail level in the applied neuron models, and thus is computationally more demanding than e.g. purely rate-based models which can neglect precise spike times. This creates challenges for conventional simulation platforms such as software simulators which numerically solve differential model equations [*Mor-*

rison et al., 2005].

One new approach which avoids the bottleneck of sequential processing of signals is to emulate neural behavior in electronic circuits, so-called neuromorphic hardware. The intrinsic parallelism of neuron circuit operation allows to scale emulated network sizes without slowing down the actual on-chip execution time. First VLSI[1] implementations of neural units go back to the 1980s [*Mead and Mahowald*, 1988; *Mead*, 1989] and today are developed and utilized by a broad community [*Serrano-Gotarredona et al.*, 2006; *Merolla and Boahen*, 2006; *Vogelstein et al.*, 2007; *Häfliger*, 2007; *Renaud et al.*, 2007; *Schemmel et al.*, 2007; *Ehrlich et al.*, 2007; *Schemmel et al.*, 2008]. Operating neuromorphic hardware systems in biological real time allows to build hybrid systems with in-vitro neural networks [*Bontorin et al.*, 2007], to implement motor controls for robotics, visual sensors [*Serrano-Gotarredona et al.*, 2006] and possibly many more embedded and low-power applications. Another approach is to scale down the utilized analog circuits and hence gain a massively accelerated operation mode compared to the emulated biological time [*Schemmel et al.*, 2007, 2008].

Within the FACETS research project [*FACETS*, 2008], different approaches are integrated to face the challenge of finding and understanding computational paradigms and self-organization principles inspired by the brain. Bio-physiological measurements, analytical and numerical modeling as well as hardware emulations are applied on the single cell and on the network level. As one member of FACETS and in contrast to most other emulations of neural networks in silicon, the Electronic Vision(s) group in Heidelberg develops highly accelerated neuromorphic devices. The achieved speedup factor allows for statistics-intensive studies of long-term development of synaptic weights, which in software simulations can be realized only with extensive computational efforts.

Still, such investigations with neuromorphic hardware require reliable adjustability of parameters and intensive preparatory studies. For the Heidelberg system, which was utilized throughout this thesis, the implemented long-term synaptic plasticity had not yet been put into operation when the presented work was taken up. The major goal of all efforts presented in this thesis is to establish the possibility of applications which exploit the hardware's synaptic self-organization mechanisms. An experiment is chosen which provides a wide range of selectable complexities of the task to be solved. Hence, the underlying problems occurring during the implementation process can be clearly identified and overcome. Consequently, each solution presented in this work represents an inevitable step towards the next one, while targeting the realization of an appropriate winner-take-all experiment.

The thesis is structured into four main parts. This introduction is followed by chapter 1, which gives a more detailed overview over the applied hardware model, the implemented plasticity mechanisms and over utilized methods and concepts. In chapter 2,

[1] Very Large Scale Integration

methods developed and experiments performed by the author are presented. In order to achieve the major goal of overcoming technical barriers in utilizing the self-organization mechanisms of the chosen neuromorphic hardware system and providing a proof of functionality, various preparatory studies are presented. This includes the investigation of membrane conductance states of hardware neuron membranes and temporal dynamics of hardware synapses, cross-inhibition setup analyses and the preparation of porting a winner-take-all experiment to the hardware. Finally, the presented results and methods are discussed and an outlook regarding the perspectives of self-organization experiments on the utilized type of hardware is given.

1 Theoretical Foundations, Materials and Methods

During this work, two different simulation back-ends were utilized, which are described in this chapter: Physically implemented, neural networks are emulated with the FACETS Stage 1 Hardware. Besides the system itself, also the software layers, required for its operation and partly developed by the Electronic Vision(s) group, are presented. For comparative and preparative software simulations, NEST [*Gewaltig and Diesmann*, 2008; *The Neural Simulation Technology (NEST) Initiative*, 2008] was used.

In order to investigate unsupervised learning, different types of synaptic plasticity as well as a biologically realistic and important neuron state in active neural networks are described. Finally, a concept for measuring properties of individual units within active networks is explained.

1.1 Modeling Spiking Neural Networks

When modeling spiking neural networks different approaches are pursued: One *bottom-up* approach is to model the cellular dynamics of neurons and synapses at the molecular level in form of ion channels and detailed multi-compartment models. Aiming at maximum detailed information, this approach entails difficulties when it is applied to a higher number of elements. One simplification done on the expense of details is to combine the most important characteristics and regard neurons as point neurons without spatial extent. Further steps of abstraction combine neurons to networks and populations.

In general, modeling of neural networks is fundamentally based on biological data. The neural behavior is described with the help of mathematical equations and can be studied analytically, by software simulations or by emulations with electronic circuits.

Each of the proposed approaches has advantages and disadvantages: Analytical treatment provides an exact formulation, but often is difficult, e.g. in case of large recurrent networks of conductance based LIF[1] neurons. Simulators normally possess full flexibility and configurability in defining neuron, synapse and network parameters. All quantities are accessible and can be determined to the desired accuracy at any time. These advantages lack in the approach to emulate neural behavior on a hardware platform. Hardware implementations are bound to one specific neuron model, which can be implemented in

[1] Leaky Integrate-and-Fire

modified form only in a subsequent revision which demands several months. Other restrictions inherent to hardware systems are limited communication bandwidths, accessibility of parameters and their possible ranges. In return, the hardware approach entails advantages that cannot be realized by software platforms. For instance, the system works intrinsically in parallel, which allows the scaling of the network size nearly without loss of performance. Regarding the power consumption of simulator platforms dealing with network sizes in the range of $10^6 - 10^7$ neurons, other advantages of the hardware system currently under development become obvious: The hardware solution is a power and space efficient implementation for emulating large scale neural networks. Thus, using neuromorphic hardware as an additional tool in modeling neuroscience is a promising approach.

1.2 Hardware System

The neuromorphic network chip utilized in this work is part of the FACETS Stage 1 hardware system [*Schemmel et al.*, 2006, 2007]. The chip was produced using a standard 180 nm CMOS[2] process and is the prototype of a large wafer-scale integration system [*Fieres et al.*, 2008; *Schemmel et al.*, 2008] currently under development.
The chip is mounted on a carrier board, which among others carries an FPGA[3] and RAM[4] modules for storing input and output data. The FPGA implements experiment and communication control. The carrier board is plugged onto a backplane, which communicates via a PCI[5] card to a host PC. The hardware framework is explained in detail in [*Grübl*, 2007; *Fieres et al.*, 2004; *Philipp et al.*, 2007]. Analog signals, like the membrane potential or reversal potentials, can be read out via an oscilloscope and accessed via a network connection. The chip contains 384 neuron circuits, and almost $5 \cdot 10^4$ synapses.

1.2.1 Hardware Neuron Model

The hardware neuron circuits implement a leaky integrate-and-fire model with conductance-based synapses, based on existing phenomenological models [*Destexhe*, 1997], with a variable speed-up factor between 10^4 and 10^5. In the presented work, the speed-up factor was set to 10^5, i.e. 10 ns in the hardware system correspond to 1 ms in biological time.

[2] Complementary Metal Oxide Semiconductor
[3] Field Programmable Gate Array
[4] Random Access Memory
[5] Peripheral Component Interconnect

The neurons' membrane potentials obey the differential equation:

$$-C_m \frac{dV}{dt} = g_l(V - E_l) + \sum_j p_j(t) g_j(t)(V - E_E) + \sum_k p_k(t) g_k(t)(V - E_I) \qquad (1.1)$$

The constant C_m stands for the total membrane capacitance. The three summands on the right hand side represent three different ion channels: The first one models the leakage, that drives the membrane to its resting potential E_l with a constant leakage conductance of g_l if no input reaches the neuron. The two remaining terms model the excitatory and inhibitory ion channels with their reversal potentials E_E and E_I, respectively. The index j of the first sum runs over all excitatory synapses, index k in the second sum over all inhibitory synapses. The conductance courses $g_E(t)$ ($g_I(t)$) are shaped as very sharp increases to a maximum value $g_{E\max}$ ($g_{I\max}$), followed by an exponential decrease with a time constant τ_{syn}. Each synapse has an open probability $p_{j,k}(t)$, which can be modified by short-term plasticity mechanisms [*Dayan and Abott*, 2001]. The short-term plasticity implementation is described in detail in [*Schemmel et al.*, 2007] and [*Bill*, 2008]. Furthermore, for every synapse $g_{\max}$ can be modified by spike-time dependent plasticity (STDP). The hardware implementation of the long-term plasticity mechanism is reported in [*Schemmel et al.*, 2007].
Once, the membrane voltage exceeds its threshold voltage V_{thresh}, the neuron will fire an action potential, then will be pulled to a reset voltage V_{reset}, remaining there for some refractory period τ_{ref}, and afterwards will follow the forces of its leakage, excitatory and inhibitory mechanisms again.

1.2.2 Limitations of the Hardware System

Some limitations of the hardware system have to be kept in mind throughout this work, because parameters for software simulations had to be chosen in a way, that allows to port the setup onto the hardware. Some major restrictions, being of importance for this work, are:

- Synaptic weights are not continuous, but quantized with a 4 bit-resolution.
- The maximum input frequency is limited to ca. 2 kHz (biological scale) for the entire network.
- Each neuron can be connected to a maximum of 256 pre-synaptic inputs.
- In the current chip revision the number of neurons from which spikes can be recorded at the same time is limited to approximately 48 neurons [*Müller*, 2008, section 4.3.2].
- Several parameters are not arbitrarily adjustable, e.g. the synaptic time constant τ_{syn}.
- Parameters can reveal parasitic interference. For instance, the control current *drvifall* can induce a serious and permanent synaptic conductance. This has a

strong impact on the resting potential of the neuron and, thus, on the spiking behavior and the available dynamic range. Furthermore, the reset mechanism and threshold potentials are adjusted via a shared parameter, called *icb*. Some neurons lack either a precise threshold voltage or a reliable reset after spike emission.

1.3 Software Framework

1.3.1 Interface Layer Stack

The interface to the FACETS hardware and to the software simulator NEST [*Diesmann and Gewaltig*, 2002; *Gewaltig and Diesmann*, 2008; *The Neural Simulation Technology (NEST) Initiative*, 2008] is organized in several layers. On top of the interface layer stack stands the meta language PyNN [*Davison et al.*, 2008], in which the whole experiment is defined. At the beginning of each experiment script the back-end specific Python [*Drake*, 2000] module is included, e.g. `PyNN.hardware` or `PyNN.nest`. The respective back-end is accessed through an additional layer PyHAL[6] [*Brüderle et al.*, 2007] for the hardware and the scripting language SLI for NEST. This layer offers access functions to the specific back-end and builds the connection between the simulator or emulator platform to the PyNN interface. In the hardware system, the access from PyHAL to the low-level code is done with the help of the Boost.Python wrapper library [*Abrahams and Grosse-Kunstleve*, 2003]. The low-level code is written in C++ and implements the configuration of the chip, communication plus sending and receiving of spike trains.

1.3.2 The Meta-Language PyNN

"PyNN is a Python package for simulator-independent specification of neuronal network models" [*Davison*, 2008]. It is an interpreter-based API[7] for describing experiments using the network models supported by the back-end [*Davison et al.*, 2008]. PyNN offers the possibility to define experiments once and execute them on different back-ends without modification. This allows to port experiments between different platforms and makes it easier to compare the results gained, e.g. to verify the hardware model.

1.3.3 The Simulator NEST

The software simulator NEST [*Gewaltig and Diesmann*, 2008; *The Neural Simulation Technology (NEST) Initiative*, 2008] is a framework for simulating large networks of biologically realistic neurons. It provides various synapse types, recording devices and neuron models and can be extended by user-written models. The neuron model, utilized in the simulations presented in this thesis, exactly implements equation 1.1 and

[6] Python Hardware Abstraction Layer
[7] Application Programming Interface

is described in detail in [*Muller*, 2006]. Just like the utilized hardware, NEST can be interfaced through the Python-based [*Drake*, 2000], simulator-independent scripting language PyNN [*Davison et al.*, 2008], allowing for a unified description and analysis of the performed experiments [*Brüderle et al.*, 2007].
Drawbacks of NEST are for example:

- Parallelization of computation needs extra effort and is limited by the number of available processors [*Morrison et al.*, 2005]. Thus, scaling network models up to large sizes slows down computation time significantly.
- The synaptic time scale parameter τ_{syn} is part of the neuron parameter set, i.e. all synapses that stimulate one neuron can only have one single τ_{syn}.

1.4 High Conductance States

The high-conductance state describes a state of single neurons within an active network. For single neurons, this particular state can be defined by the condition that the total synaptic conductance received by the neuron is larger than the leakage conductance. There is experimental evidence for the high-conductance state in in-vivo cells, e.g. in awake and attentive animals [*Destexhe et al.*, 2003; *Boustani et al.*, 2007], or in-vitro in localized sub-populations [*Cossart et al.*, 2003].

Under increased synaptic stimulation, cortical neurons exhibit a high total membrane conductance. This results in a shorter membrane time constant which has wide consequences on the behavior of individual neurons and networks of neurons. Characteristically for neurons in the high-conductance state is a low input resistance, depolarized membranes with large membrane potential fluctuations, dominant inhibitory conductances and a stochastical response to a given stimulus due to fluctuating background activity [*Destexhe et al.*, 2003]. Other consequences for a cell in the high-conductance state due to increased background activity is the attenuation of distal somatic inputs on the somatic tree, an increased sensitivity to temporal synchrony or, in other words, an increased temporal resolution capability (see [*Koch*, 1999; *Destexhe et al.*, 2003; *Rudolph and Destexhe*, 2006], section 2.1). Thus, the conductance state of a cell has a strong effect on its spatio-temporal processing properties. It is supposed that the transition from a low to a high-conductance state "is accompanied by a switch from encoding time-averaged input with firing rate to encoding transient inputs with precisely timed spikes." [*Prescott et al.*, 2006].

The high-conductance state is a feature inherent to the utilized conductance based

neuron model. Equation 1.1 can be rewritten as follows:

$$-C_m\frac{dV}{dt} = \left(g_l + \sum_j p_j(t)g_j(t) + \sum_k p_k(t)g_k(t)\right)\cdot V \\ -g_l E_l - \sum_j p_j(t)g_j(t)E_E - \sum_k p_k(t)g_k(t)E_I \tag{1.2}$$

Or, in a shorter form by neglecting short term plasticity effects $p_{j,k}(t)$:

$$-\frac{dV}{dt} = \tau_m(t)^{-1}V - \frac{g_l E_l - \sum_j g_j(t)E_E - \sum_k g_k(t)E_I}{C_m} \tag{1.3}$$

with the effective membrane time constant $\tau_m(t) = \frac{C_m}{g_L+g_E(t)+g_I(t)}$.

The total membrane conductance $g_T(t) \equiv g_L + g_E(t) + g_I(t)$ strongly influences membrane dynamics by affecting the shape and amplitude of post-synaptic potentials (PSPs). Thus, one important property of high-conductance states is the reduction of the membrane time constant τ_m, which makes neurons follow their input fluctuations immediately.

Hence, as derived in [*Shelley et al.*, 2002], equation 1.1 can also be written as

$$\frac{dv}{dt} = -\tilde{g}_T(t)[v(t) - V_s(t)] \tag{1.4}$$

with the so-called effective reversal potential $V_s(t) = (\tilde{g}_E(t)\tilde{V}_E - \tilde{g}_I(t)|\tilde{V}_I|)/\tilde{g}_T(t)$, where $\tilde{g}_T(t)$, $\tilde{g}_E(t)$ and $\tilde{g}_I(t)$ are the total, the excitatory and inhibitory conductances at time t divided by the membrane capacitance C_m, and $\tilde{V}_E$, $\tilde{V}_I$ are normalized, dimensionless transformations of the excitatory and inhibitory reversal potentials.
In a high-conductance state, the mean value of $g_T(t)$ is large compared to g_L because of large synaptic conductances. From the equation above follows that in a high-conductance state, a large difference between $v(t)$ and $V_s(t)$ causes a large change of $v(t)$. Thus, in such a state, the subthreshold membrane potential is well approximated by the effective reversal potential:

$$v(t) \simeq V_s(t)$$

In that way, the modulation of $V_s(t)$ and consequently $v(t)$ is approximated by the ratio of excitatory to inhibitory conductances, (see [*Shelley et al.*, 2002], eq. (16)).

1.5 Synaptic Plasticity

Synaptic plasticity is supposed to be one mechanism for learning and memory formation. Two kinds of plasticity operating on different time scales can be distinguished: short-term plasticity (STP) and long-term plasticity (LTP). Both plasticity mechanisms can

have potentiating and depressing effects on synapses. Short-term plasticity affects the synaptic efficacy on time-scales of a few milliseconds to seconds [*Markram et al.*, 1998] and does not induce permanent changes in the neural network structure. STP is supposed to be an important mechanism to stabilize the activity of recurrent networks consisting of inhibitory and excitatory neurons [*Sussillo et al.*, 2007].

Long-term plasticity induces changes in the neural network structure by the change of synaptic weights for varying amounts of time, generally persisting tens of minutes or longer. The first theoretical prediction goes back to Donald Hebb, who stated in 1949: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." or in other words: "What fires together, wires together."

First approaches to model long-term plasticity in spiking neural networks were based on firing rates [*Hebb*, 1949]. The derived learning rule is not stable, i.e. synaptic weights can grow unconstrained and no competition between synapses exists. One prominent extension to Hebb's learning rule is the Bienenstock-Cooper-Munro rule [*Bienenstock et al.*, 1982] which stabilizes the weight development, induces competition and is experimentally motivated. Another extension is Oja's learning rule which additionally features the normalization of synaptic weights and can implement a principle component analysis of the input [*Oja*, 1982]. This rule is rather based on theoretical considerations than on experimental data.

Experimental work showed that the precise timing of input and output spikes plays a crucial role in the weight modification in hippocampus [*Levy and Steward*, 1983; *Bi and Poo*, 1997], leading to the concept of spike-time dependent plasticity (STDP) [*Song et al.*, 2000; *Dan and Poo*, 2004; *Legenstein et al.*, 2005; *Morrison et al.*, 2007, 2008]. Furthermore, there have been extensive studies on the cellular mechanisms of STDP in the past decades, which revealed that synaptic plasticity is not only dependent on the precise spike-timing, but also on the post-synaptic concentration of calcium ions, the dendritic location, the connection type, neuromodulators, inhibitory activity and the integration of complex spike trains [*Caporale and Dan*, 2008]. In contrast to phenomenological models, biophysical models try to explain the biochemical and physiological processes leading to synaptic modifications. As synaptic plasticity can have different forms, depending on the cortical region, the involved cell types, the location of involved cells in the layered structure and even the type of the presynaptic neuron, biophysical models try to account for experiments within a single system and might not be valid for a different system.

Based on phenomenological models described in [*Bi and Poo*, 1997; *Dayan and Abott*, 2001], the following weight update function is implemented in each hardware synapse [*Schemmel et al.*, 2007]:

$$F(\Delta t) = \begin{cases} A_+ exp(\frac{-\Delta t}{\tau_+}) & \text{if} \quad \Delta t > 0 \\ -A_- exp(\frac{\Delta t}{\tau_-}) & \text{if} \quad \Delta t > 0 \end{cases} \tag{1.5}$$

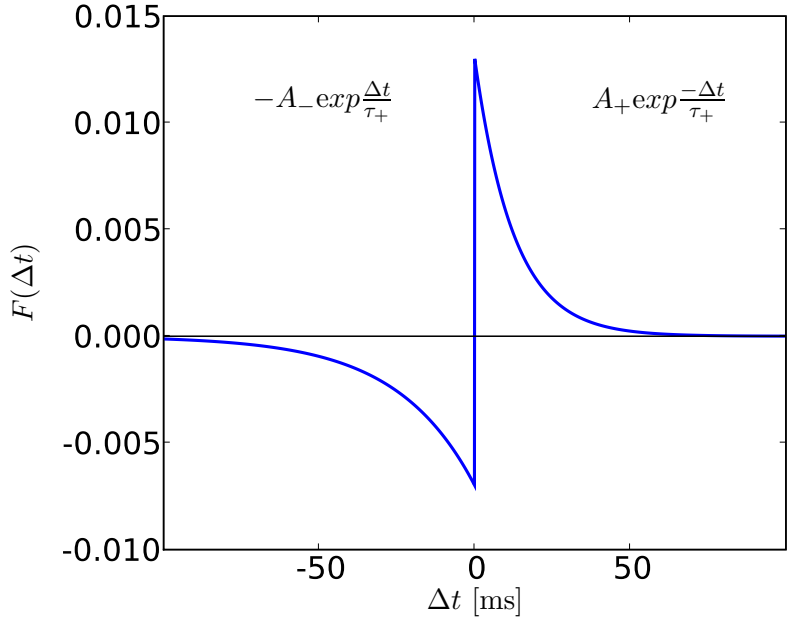


Figure 1.1: Weight update function for synapses inducing post-synaptic firing in dependence of the temporal difference between the post- and the pre-synaptic spike: $\Delta t = t_{\text{post}} - t_{\text{pre}}$

The dependence of weight modification on the temporal difference $\Delta t = t_{\text{post}} - t_{\text{pre}}$ is clarified by figure 1.1.

In the hardware system, values of the modification function $F(\Delta t)_{c,a}$ are accumulated separately for causal and acausal firing on two capacitors storing values ΣF_c, ΣF_a. These two values are periodically read out and checked with respect to two aspects:

$$\begin{aligned} &a) \quad |\Sigma F_c - \Sigma F_a| > F_{\text{threshold}} \\ &b) \quad \Sigma F_c - \Sigma F_a > 0 \end{aligned} \tag{1.6}$$

If a) is true, the 4-bit weight is updated according to a specified lookup table. There are two lookup tables, one for the case that the causal weight modification function is larger, i.e. equation 1.6 b) is true, and one for the other case. Both lookup tables can be arbitrarily programmed with 16 4-bit values which determine the new weight for every possible old weight. After the weight update took place, both values $\Sigma F_{c,a}$ are set to zero for the respective synapse.

For each row (representing one axon of a neuron) within a synapse array, the STDP mechanism can be enabled individually. The period of reading out $\Sigma F_{c,a}$ depends on the number of rows using STDP. If STDP is activated for all rows, the maximum period

between two updates is ca. 456 μs on chip, which corresponds to 45.6 seconds in biological time. This constrains the range for biological realistic setups and is a big difference compared to simulator tools, which process weight modifications immediately, or with a delay of one spike.

The threshold for the accumulated weight modification functions F_{thresh} is a crucial parameter, that has to be adjusted according to the readout period update. If $F_{\text{threshold}}$ in equation 1.6 is small, but the readout period is large compared to the mean interval between correlations, many correlations stay without effect on the weight. On the other hand, when $F_{\text{threshold}}$ is large, the weight change due to accumulated effects of necessarily many correlated spike pairs of one type (either causal or acausal) is applied late. For some kinds of learning experiments this might be too late.

1.6 Spike-Triggered Averaging

In both biological and hardware emulated neurons, noise influences the membrane potential of individual cells. Thus, it is in general not possible to extract detailed information from the recorded raw data. One method to extract the effect of a single action potential on the membrane potential is spike-triggered averaging (STA) [*de Boer and Kuyper*, 1968; *Matsumura et al.*, 1996].

In the STA procedure, the membrane potential of a single neuron is recorded, and points in time where a pre-synaptic action potential occurs are stored as trigger signals. The neuron's voltage trace samples surrounding the trigger signals are summed up and divided by the number of stored trigger signals to get a so-called spike-triggered average. Provided that averaging is done with sufficient statistics, the resulting time course of the membrane potential is almost free from noise and reveals the shape of a PSP[8]. Figure 1.2 shows an example of an EPSP[9] which was acquired by recording ca. 2500 samples on a VLSI neuron.

For the synapse model and the applied synaptic parameters used throughout this thesis, the synaptic time constant τ_{syn} can be measured indirectly by fitting an exponential decay to an STA trace acquired from a neuron in the high-conductance state, because the low-pass filtering effect of the membrane can be assumed to be significantly smaller than τ_{syn}.

[8] Post-Synaptic Potential
[9] Excitatory Post-Synaptic Potential

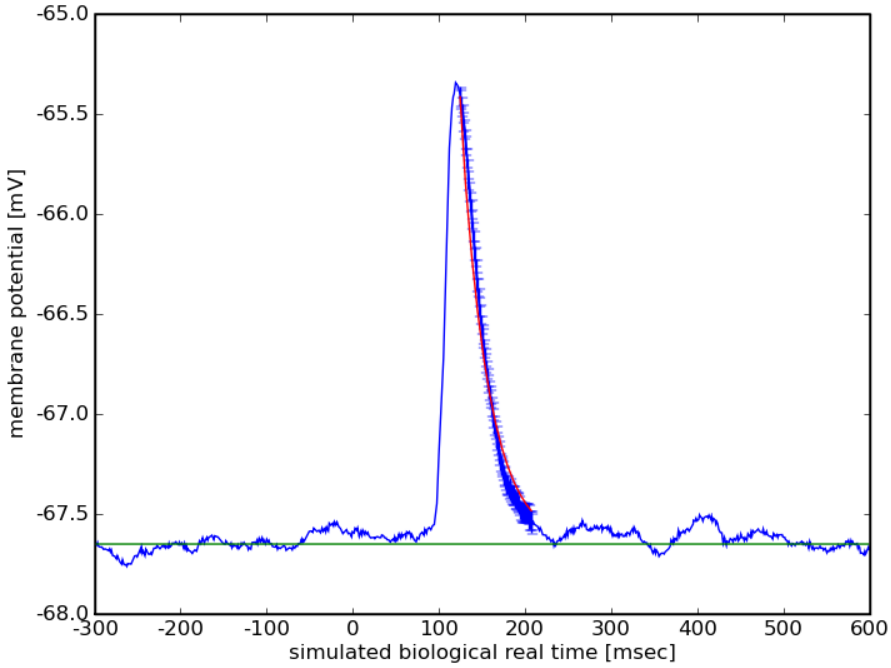


Figure 1.2: Post-synaptic potential gained through spike-triggered averaging over ca. 2500 samples recorded from a hardware neuron. Voltage and time-scale are converted to biological scale.

2 Experiments and Results

The FACETS Stage 1 hardware system is a prototype for a wafer-scale system [*Schemmel et al.*, 2008]. Since the current state of the hardware system does not allow to use essential features like STDP or the recording of arbitrary many neurons, the following sections present preparatory studies for the application of long-term learning experiments on the hardware system. In order to investigate the practicability of a self-organized winner-take-all based on mutual inhibition and STDP, these setups are studied with the help of software simulations in section 2.3 and 2.4. One of the acquired results shows the importance of synaptic time constants τ_{syn}. As the conductance time course and with this the synaptic time constants are not directly accessible in hardware, a high-conductance state has to be induced, which allows to estimate the desired quantity τ_{syn} from the membrane potential. Thus, a high-conductance state test which allows the separation of conductance states is essential. In a first step, a spike-based test concept is presented, verified through software simulations and applied on the hardware.

2.1 A Spike-based High Conductance State Test

Remark: This section contains segments of a paper draft written by the author and Daniel Brüderle.

2.1.1 Motivation

Neural behavior can be characterized by a variety of dimensions, e.g. spike rate, membrane potential traces, currents and conductances, amongst others. Depending on the observed system, it can be necessary to deduce unaccessible or with difficulty measurable magnitudes from easily accessible ones. E.g. for in-vivo and in-vitro recordings, the strength of a synaptic connection typically has to be deduced from the correlation of spiking and membrane activity of its pre- and post-synaptic neurons [*Bi and Poo*, 1997; *Dan and Poo*, 2004]. In general, models of neuronal and synaptic dynamics often involve variables which are hard to observe in-vivo or in-vitro.

Software simulators have full flexibility in accessing variables involved in the respective model. In contrast, the utilized neuromorphic hardware does not offer the possibility to access some variables, as for example currents and conductances. Thus, they have to be deduced from accessible observables. The accessible observables in the hardware system are its spike output, membrane potential traces and possibly evolving synaptic

weights. The neuromorphic system design has been optimized for the access to all action potentials generated during an experiment via a fast digital connection. Since the subthreshold membrane potentials have to first be acquired via an oscilloscope connected to the hardware and then need to be integrated into the operating and evaluating software via TCP/IP sockets [*IEEE*, 2004; *Braden*, 1989], this acquisition channel is rather slow and inefficient. Thus, a deduction of hidden variables like conductances from nothing but the spike output is highly desirable.

The presented spike-based high-conductance state makes it possible to distinguish different neuronal conductance states. But why is it important to distinguish between different conductance states?

As will be needed for investigations described later within this thesis, it is essential to excite a high-conductance state in order to deduce the synaptic time constants from the membrane potential time course with the help of spike-triggered averaging. The role of synaptic time constants is, amongst others, discussed in section 2.3.5.

Furthermore, for future biologically realistic experiments on neuromorphic systems, but also for basic specification of hardware subunits, finding a working point in the high-conductance state is essential.

Plus, as described in section 1.4, the conductance state of a neuron has strong impact on its integrative properties. Depending on its conductance state, a neuron either acts as an integrator (in low-conductance states when the membrane time constant is small) or as a coincidence detector (in a high-conductance state when the membrane time constant is short). Thus, a change of the conductance state means a change in the coding mechanism in neural information processing.

In the following, the effects of synaptic contributions to the total membrane conductance - in conjunction with the correlation of the applied input spike trains - on the output spike rate are studied. A purely spike-based and hence hardware-compatible method to estimate the amount of necessary synaptic stimulation in order to operate within a high-conductance state is presented. Differences and hardware specific advantages of this method compared to a similar one introduced in [*Rudolph and Destexhe*, 2006] will be discussed in section 2.1.5.

2.1.2 Concept and Setup

Test Concept

As the membrane time constant is dependent on the total membrane conductance, one can use membrane dynamics to estimate the conductance state of a neuron. When the total conductance is large, τ_m is small and thus enables the neuron to react immediately to changes in the input conductance. A small time constant can cause the neuron to work as a coincidence detector, i.e. it allows the neuron to resolve dense input signals because it responds to different input signals independently. Thus, the basic idea of the

proposed high-conductance state test is to estimate the total membrane conductance of a neuron by its ability to separate excitatory post-synaptic potentials (PSPs) which are temporally close. The integration of successive PSPs on a membrane is less likely to cause an action potential if the temporal course of these PSPs is shorter. Assuming fixed time constants for the input-triggered increase and decrease of $g_E(t)$, the shape of the resulting PSP is shortened or stretched by the total membrane conductance.

Consequently, compared to a low-conductance state, in a high-conductance state successive input spikes have to be temporally closer to cause an increase in the output firing rate, which can be regarded as a better time resolution property of the neuron.

In other words, the low-pass filter property of the membrane determines its quality as a coincidence detector. This makes it possible to deduce the total membrane conductance merely from input and output spike data. Figure 2.1 illustrates the effect of different total membrane conductances on the superposition of PSPs. The same sequence of spikes - a single spike followed by a quadruple - arrives at a relatively slow (green line) and at a fast (blue line) membrane. Due to the resulting different temporal courses of the PSPs, those on the slow membrane add up to a larger effective amplitude compared to those on the fast membrane.

Test Setup

For testing the input driven responsiveness of a membrane, a single neuron with a constant leakage conductance g_l is utilized. In order to vary the externally driven component of the membrane conductance, it receives a set of Poisson spike trains through N_E excitatory and N_I inhibitory synapses. Each spike train has the same firing rate ν_{in}. The decay time constants τ_{syn} and the maxima $g_{E\max}$ and $g_{I\max}$) for the excitatory and inhibitory conductances are kept constant during all experiments. The major aim of the test is to find an average synaptic conductance $\overline{g_{\text{syn}}} \equiv \langle g_E(t) + g_I(t) \rangle$ which results in a high-conductance state. With given values for τ_{syn}, $g_{E\max}$ and $g_{I\max}$, the temporal integration over N_E excitatory and N_I inhibitory spike trains with firing rate ν_{in} leads to the following total average synaptically induced conductance:

$$\overline{g_{\text{syn}}} = \tau_{\text{syn}}\, \nu_{\text{in}}\, (N_E\, g_{E\max} + N_I\, g_{I\max}) \qquad . \tag{2.1}$$

The conductance state of a neuron stimulated by pure Poissonian spike trains can be changed by

- the synaptic weights $g_{\text{exc}}, g_{\text{inh}}$,
- the synaptic time constant τ_{syn},
- the number of inputs stimulating the neuron $N_E + N_I$,
- the leakage conductance g_{leak},
- the input rate ν_{in}.

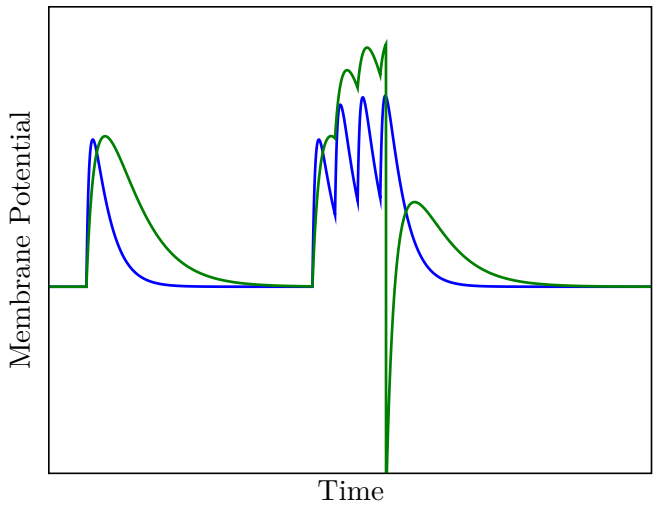


Figure 2.1: NEST Simulation: Overlapping PSPs on a membrane with high (blue) and low (green) total conductance. For this schematical example, the membrane conductance is adjusted by changing its leakage. The PSPs result from conductance time courses with identical decay times. The conductance amplitudes for both the high and the low conductance membrane have been adjusted such that the amplitudes of the single PSPs become the same in both cases. This allows for an illustrative different maximum amplitude of four overlapping PSPs triggered by identical input spikes. For the fast, i.e. higher conductance membrane, the accumulated potential is not high enough to reach the spike threshold, while for the slow one it is.

The synaptic weights are not used to control the input conductance, because the impact of individual spikes strongly increases with $g_{\mathrm{exc}}, g_{\mathrm{inh}}$. This would lead to larger membrane fluctuations which is on one hand side characteristic for a high conductance state, but on the other hand imposes two problems: First, the probability to spike is increased which would lead to a higher output rate which is not desired as described above. An increase of the firing threshold as circumvention of this would not solve this problem, because the membrane voltage of the hardware neuron would tend to leave the dynamic range of the neuron circuit and thus lead to false behavior.

The leakage conductance of a neuron can also not be used to change the conductance state, because in the hardware system this parameter has to be used in a calibration routine to adjust the membrane time constant. This results in a very limited remaining available range for g_{leak}.

As the number of inputs in the hardware system is limited to 256, it is not possible to use this parameter to investigate the membrane dynamics over a large range of input conductance.

Consequently, to control $\overline{g_{\mathrm{syn}}}$, the input frequency ν_{in} is varied. If all other parameters remain constant, this would result in a corresponding variation of the average membrane potential and of the output firing rate. As the information gained by the proposed method is based on the rather small changes of the output rate as a function of input correlations, a variance in the output rate due to changed Poisson background stimulation deteriorates the underlying signal which is intended to be detected. Thus, a variance of the output rate due to enhanced stimulation with Poisson spike trains results in a decreased strength in the desired output signal.

An increase in the output rate is also not preferable for the following reason: A high output rate affects the responsiveness of the membrane by clamping the membrane potential to V_{reset} for the refractory period τ_{ref} and making it not responsive to any input. Consequently, an increase of the output rate due to Poisson background stimulation is not desired.

In order to circumvent these undesired correlations between the average membrane potential, the output rate and the responsiveness of the membrane, the output rate is kept within a limited range ($\nu_{\mathrm{out}} = \nu_{\mathrm{target}} \pm 0.25\nu_{\mathrm{target}}$) by changing the ratio N_E/N_I, whereas the sum $N_E \cdot g_{E\max} + N_I \cdot g_{I\max}$ remains constant. This is useful, because in that way the total membrane conductance can be changed continuously and well controlled by simply varying the input rate ν_{in}.

The number of excitatory and inhibitory inputs are adjusted such that the output rate due to Poisson background stimulation does not exceed the limit of (4.0 ± 1.0) Hz.

In addition to the Poisson type background of ν_{in}, a test stimulus is injected into the neuron via n_{test} excitatory synapses, which consists of periodic packages of n_{pack} equidistant spikes. The period T_{PP} from package to package is kept constant, while the inter-spike interval T_{ISI} within one package is varied from 0 ms to $T_{\mathrm{ISI,max}} \equiv \frac{T_{\mathrm{PP}}}{n_{\mathrm{pack}}}$. With

$T_{\mathrm{ISI}} = T_{\mathrm{ISI,max}}$, one package exactly fills T_{PP}. This approach ensures that the total spike rate fed into the neuron through the test synapses is constant, independently of T_{ISI}. Figure 2.2 exemplary illustrates the test setup: a neuron receives input from Poisson spike trains of a certain frequency ν_{in} (only a subset is shown) and additionally from a test stimulus consisting of the packages of equidistant spikes. When the periodic spike packages arrive, the output rate temporarily increases.

The mean output rate is dependent on T_{ISI}, because shorter time intervals lead to stronger accumulation of post-synaptic potentials (PSPs) on the membrane. But as discussed above, for a constant T_{ISI} the output rate also depends on the total membrane conductance respectively on the width of a PSP. Hence, sweeping T_{ISI} for various values of $\overline{g_{\mathrm{syn}}}$ (regulated through ν_{in}) and measuring the resulting output firing rate will result in different response curves showing the temporal resolution capability of the neuron.

In the following, all output rates indicated with an f actually represent the difference between the output rate acquired with a specific test stimulus configuration minus the output rate with no test stimulus at all, $f(T_{\mathrm{ISI}}) \equiv \nu_{\mathrm{out}}^{\mathrm{stim}}(T_{\mathrm{ISI}}) - \nu_{\mathrm{out}}^{\mathrm{nostim}}$. This is done, because only the response to the test stimulus is of interest, not the response to the output rate caused by the Poisson background. The background determines the conductance state and thus the responsiveness of the neuron, whereas the test stimulus is needed to get quantitative information about this neuron state.

To be able to distinguish the output response curves for different conductance states, a characterizing critical quantity τ_{res} is defined as follows:

τ_{res} is the critical time interval $T_{\mathrm{PP,\,crit}}$ between test spikes at which the output rate falls below a certain threshold frequency,

$$f_{\mathrm{thresh}} \equiv f_{\mathrm{min}} + \mathrm{e}^{-1}\,(f_{\mathrm{max}} - f_{\mathrm{min}}) \qquad . \tag{2.2}$$

Here, f_{min} is the minimum output rate, i.e. the saturation rate which is reached from a certain value of $T_{\mathrm{ISI,\,max}}$ and not under-run no matter if T_{ISI} gets larger. f_{min} is defined as the mean output rate for inter-spike intervals of the test stimulus T_{ISI} in $[150\,\mathrm{ms},\ 250\,\mathrm{ms}]$. The maximum output firing rate resulting from closely coincident input spikes is f_{max}.

Figure 2.3 shows the defined quantities in a response curve gained with a background stimulation of $\nu_{\mathrm{in}} = 2\,\mathrm{Hz}$. Also shown in the plot is the $T_{\mathrm{PP,crit}}$ where the curve crosses f_{thresh}, which, according to the method proposed here, defines the temporal resolution capability of the neuron τ_{res}. An error estimate for τ_{res} is described and illustrated in figure 2.4.

By the usage of the above defined threshold output rate f_{thresh}, different conductance states can be studied quantitatively by comparing the corresponding τ_{res} for different background frequencies ν_{in}. The intention is to find an estimate for the minimum amount of input conductance which leads to a significant increase of the temporal resolution capacity of the neuron. This then can be understood as a transition from a low-conductance to a high-conductance state.

2.1.3 Software Simulation Results

In order to avoid that hardware-specific behavior might wrongly indicate the functionality of the proposed method, it is first verified qualitatively utilizing the software simulator NEST. The NEST neuron model and the parameter values are chosen to optimally resemble the hardware. Still, quantitatively equal results from hardware and software are not to be expected, since subtle non-linear parasitic hardware effects (see e.g. [*Bill*, 2008] are not included in the NEST model. Some of them will be discussed in the hardware result section.

If not explicitly stated differently, the basic set of parameters applied for the software runs is the following:

Neuron		
C_m	0.2	nF
V_{reset}	-80.0	mV
E_I	-75.0	mV
E_l	-70.0	mV
V_{thresh}	-57.0	mV
E_E	0.0	mV
g_l	2	nS

Synapses		
τ_{syn}	30	ms
$g_{E\max}$	0.4	nS
$g_{I\max}$	1.6	nS
g_{stim}	$2 \cdot g_{E\max}$	

The leakage conductance g_l has been chosen particularly low in order to have a slow membrane for the unstimulated case. If applicable on the hardware system, the remaining parameters were chosen according to [*Muller*, 2006], i.e. according to biologically realistic models. However, in a few cases the values were chosen in order to better fit the hardware system. For instance, the choice of rather large synaptic time constants was imposed by hardware limitations, because the chosen speedup factor for the hardware system does not support shorter ones. Furthermore, in order to provide the necessary amount of total synaptic stimulation, the maximum synaptic conductances are rather large, since the number of synapses to a hardware neuron is limited. It also has to be noted that neither g_l nor C_m are directly measurable for the hardware. Still, the time constant of the hardware membrane under no stimulation, $\tau_{m,\text{rest}} \equiv \frac{C_m}{g_l}$, can be easily measured. By varying a steering current that controls the not directly measurable value of g_l, $\tau_{m,\text{rest}}$ can be calibrated to the desired value.

To find the membrane time resolution measure τ_{res}, the test stimulus was applied with a package period of $T_{\text{PP}} = 1000\,\text{ms}$. Each package had $n_{\text{pack}} = 4$ spikes, with T_{ISI} being varied from 0 ms to 250 ms.

Figure 2.5 shows the response curve for a background Poisson rate of $\nu_{\text{in}} = 2$ Hz, fed into $N_E = 36$ excitatory and $N_I = 33$ inhibitory synapses, and another response curve for a background rate of $\nu_{\text{in}} = 14$ Hz, fed into $N_E = 32$ excitatory and $N_I = 34$

inhibitory synapses.

The plot shows the expected decrease in the output firing rate with growing T_{ISI} due to decreasing overlap of PSPs in the test stimulus. Every data point represents the mean value from 250 runs with 10 seconds of simulated time each, the error-bars denote the standard error of means (SEM). Also shown in the plot is the respective time resolution measure τ_{res}, which reaches different values for different background frequencies. The dependency of τ_{res} on the frequency of the Poisson background ν_{in} is subject to the following analysis.

Background Activity Increases Membrane Time Resolution

The membrane time resolution τ_{res} has been evaluated for various Poisson background rates ν_{in}. Figure 2.6 shows τ_{res} as a function of ν_{in}, exhibiting a dependency as expected. With increasing synaptic contribution to the total membrane conductance, the temporal resolution capability increases rapidly (i.e. τ_{res} decreases) until an input frequency of about $\nu_{\mathrm{in}} = 10$ Hz is reached. For higher frequencies, τ_{res} stays within the same range of approximately (20 ± 8) ms.

The rather large fluctuation of the presented data points with their mostly small errors can be explained as follows. For changing input frequencies, the τ_{res} estimation is acquired for varying output rates, since the output rate balancing via adjustment of N_E and N_I can not work perfectly. This is a drawback of the hardware neuron, which has only few synaptic inputs. This coarse granularity limits the possible performance of the algorithm that adjusts the number of excitatory and inhibitory inputs in order to keep the sum $N_E g_{\mathrm{exc}} + N_I g_{\mathrm{inh}}$ constant.

Nevertheless, the presented method still allows to distinguish between different conductance states and gives an estimate for the amount of synaptic stimulation which is required to significantly decrease the low-pass characteristics of the membrane.

Figure 2.7 shows the same experimental data plus the results of another measurement series with $\tau_{\mathrm{syn}} = 20$ ms (green curve) instead of 30 ms (blue curve). Here, the x-axis does not show the input firing rate, but the average total conductance normalized to the pure leakage conductance. In [*Brette and Gerstner*, 2005], a value of 5 for this ratio is suggested as the threshold to high-conductance states. This separating value is indicated in the plot by a dashed line and fits well the measured data, i.e. the suggested high-conductance regime clearly differs from the low-conductance states in terms of membranic time resolution capabilities.

2.1.4 Hardware Implementation

As indicated above, variations from the pure software results are to be expected, since the hardware is subject to electronic phenomena like noise, crosstalk and leakages. Especially the leakages in hardware introduce some dynamics (see e.g. [*Bill*, 2008]) which are

hard to be mapped to the available standard neuron models in NEST. E.g., the circuits which generate the synaptic conductance courses are assumed to exhibit weight dependent leakage conductances towards their corresponding reversal potentials. Furthermore, these leakages are often temperature dependent plus subject to process variations during production of the chip, thus being hard to quantify. Compensation methods are under development and minimization is an aim of planned chip revisions, but these approaches could not yet be considered for this study. Thus, compared to the software simulation, a positive total membrane conductance offset has to be expected.

A comparison between the figure showing the time resolution capability of a hardware neuron (see figure 2.8) and the software simulations show, that also for the hardware neuron τ_{res} decreases rapidly with increasing background stimulation. This is an evidence that the generation of a high conductance state can be reached with a moderate background stimulation fitting well the limitations of the hardware system.

2.1.5 High Conductance State Test: Discussion and Conclusion

A purely spike-based method for the measurement and description of conductance states of a neuron's membrane has been presented. The concepts have been shown to work properly both in a software simulation study and in a hardware implementation. For the hardware substrate, this method is an essential key to access information about the total membrane conductance and thus to the synaptic temporal dynamics, which are both not directly accessible. Low-pass filtering effects of the membrane can be minimized to an acceptable level in case a high-conductance state can be asserted.

A similar method described in [*Rudolph and Destexhe*, 2006] operates also spike-based, but has a few draw-backs compared to the presented one - mainly due to portability issues for the hardware platform, which does not offer the flexibility for artificial setups as software simulators do. First, the method in [*Rudolph and Destexhe*, 2006] requires in the order of 30 synaptic inputs which synchronously generate very strong conductance courses. Since the hardware platform is very limited in terms of both number of synaptic inputs and - at least in practice - the available range and reliability of synaptic efficacies, this setup is not well suited for being adopted. Furthermore, the sending of more than 4 perfectly synchronous spikes into one neuron at a time is not possible on the utilized chip. Sweeping arbitrary input firing rates is not possible either, the proposed ranges of up to 150 Hz clearly exceed the hardware bandwidths.

A more general advantage of the method proposed here is the fact that the suggested measure is given in millisecond dimension and thus gives a more intuitive quantity for a neuron's temporal resolution property. Furthermore, the method proposed in [*Rudolph and Destexhe*, 2006] probably is more difficult to be automatized due to the fact that it requires the detection of peaks in inter-spike interval histograms which often are ambiguous. The algorithm needs to find the height of a peak that is not necessarily the global, but just a local maximum within the histogram, and can be determined only by

its expected location.

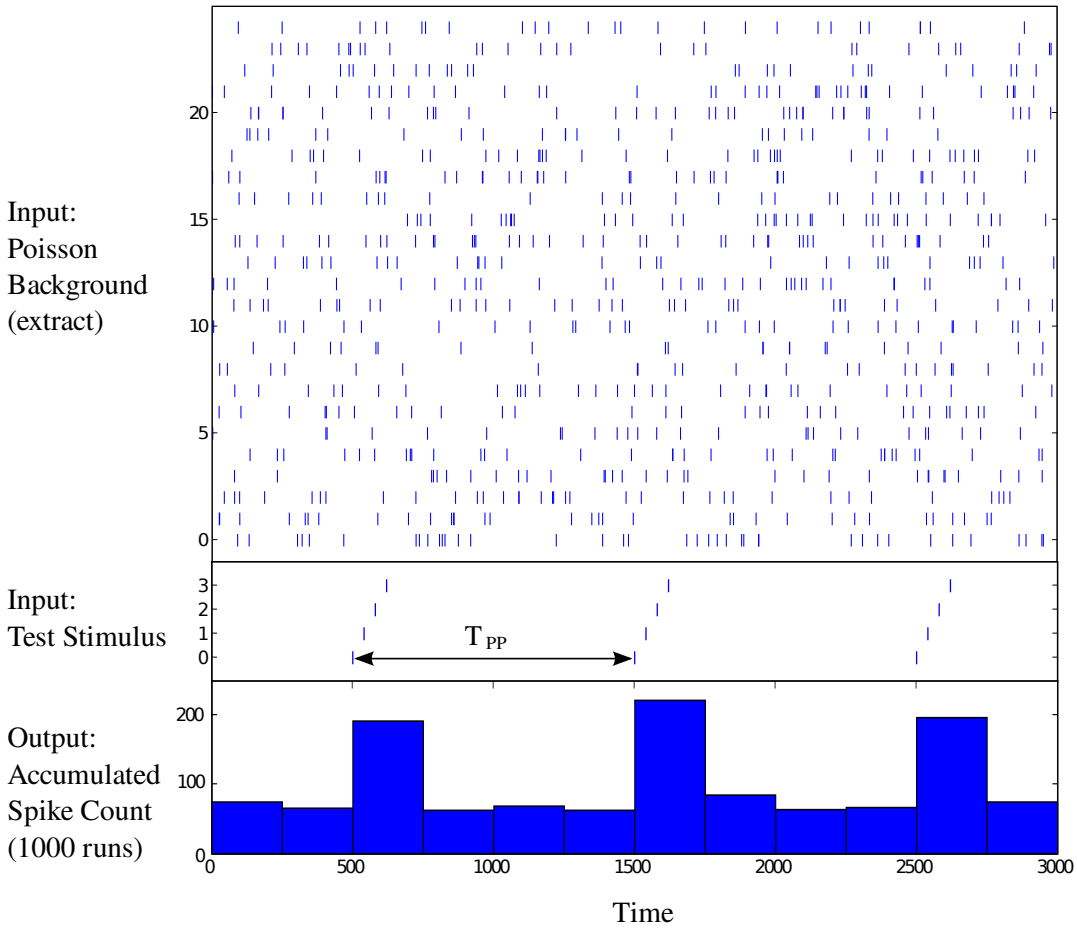


Figure 2.2: NEST Simulation: Example of the spike-based method for membrane time resolution evaluation. Shown are input and output of a neuron under test. Top: Raster plot of parts of the Poisson background with $\nu_{\text{in}} = 10\,\text{Hz}$. Middle: Test stimulus fed into the neuron. Bottom: Resulting output spike count accumulated over 1000 runs.

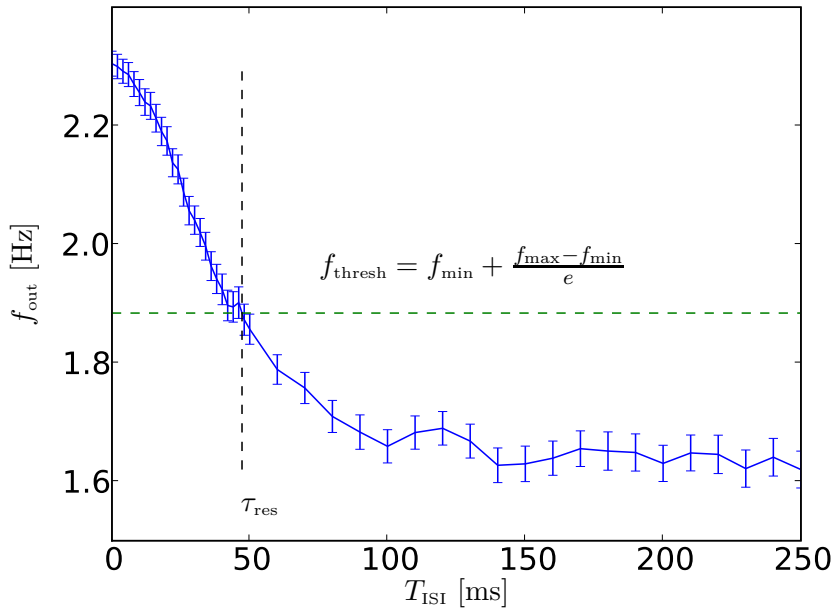


Figure 2.3: NEST simulation: Output firing rate $f_{\text{out}} = \nu_{\text{out}}^{\text{stim}} - \nu_{\text{out}}^{\text{nostim}}$ vs. inter-spike interval T_{ISI} of test spikes. The dashed line shows f_{thresh} defined as $f_{\text{thresh}} = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}})/e$. f_{min} is defined by the mean output rate for T_{ISI} in [150 ms, 250 ms]. The shown output rates are mean values averaged over 250 simulation runs with 10 s duration each, error bars are standard error of the mean. The vertical line marks the critical package density $T_{\text{PP, crit}}$, where the output rate falls below f_{thresh}. This critical package density is in the following named τ_{res}, because it is a measure for the temporal resolution capability of the neuron.

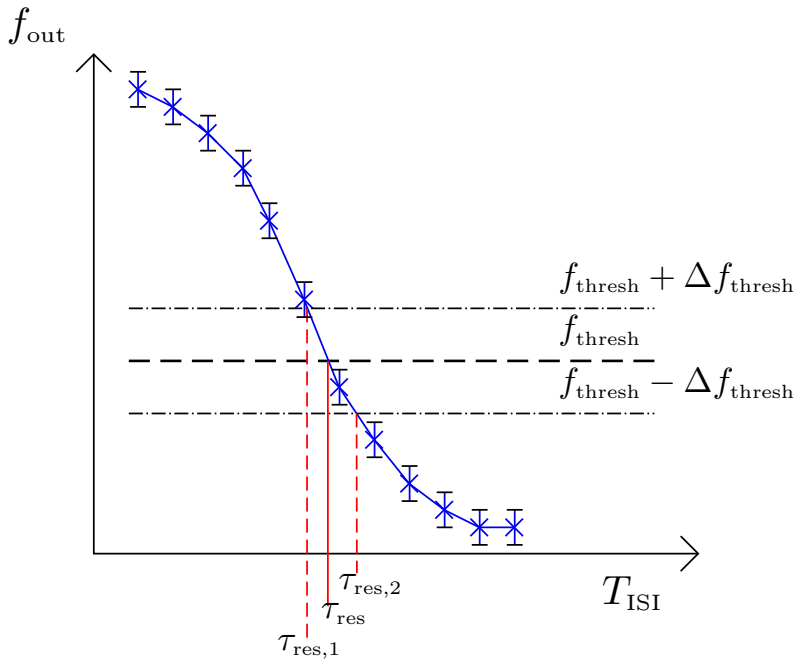


Figure 2.4: Schematic for error estimation of τ_{res}. x-axis: inter-spike interval of test stimulus package. y-axis: f_{out}. As f_{thresh} (black dashed line) is determined from two quantities with an error in measurement, f_{thresh} is also prone to an error Δf_{thresh} (black dash-dotted lines): $\Delta f_{\text{thresh}} = \sqrt{(\partial f_{\text{thresh}}/\partial f_{\text{min}} \cdot \Delta f_{\text{min}})^2 + (\partial f_{\text{thresh}}/\partial f_{\text{max}} \cdot \Delta f_{\text{max}})^2}$. Δf_{max} is the standard error of the maximum mean value of the response curve. f_{min} is mean value of f_{out} between T_{ISI} =150 ms and 250 ms. Δf_{min} is the standard deviation of that mean value. The response curve f_{out} vs. T_{ISI} crosses the error bars $f_{\text{thresh}} \pm \Delta f_{\text{thresh}}$ at two points. The x-coordinate of these points are marked by $\tau_{\text{res, 1/2}}$ (red dash-dotted lines). The differences $\tau_{\text{res}} - \tau_{\text{res,2}}$ and $\tau_{\text{res}} - \tau_{\text{res, 1}}$ are in the following the errors of the temporal resolution capability of a neuron at a certain input frequency.

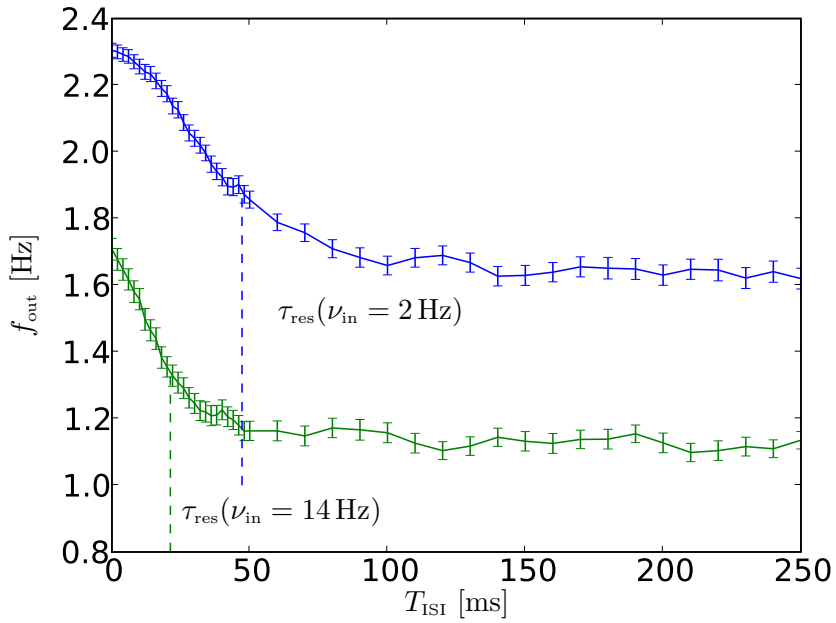


Figure 2.5: NEST simulation: Comparison of two output firing rates $f_{\mathrm{out}} = \nu_{\mathrm{out}}^{\mathrm{stim}} - \nu_{\mathrm{out}}^{\mathrm{nostim}}$ vs. inter-spike interval T_{ISI} of test spikes. The blue curve shows the response of a neuron that was stimulated with Poisson background of $\nu_{\mathrm{in}} = 2$ Hz, the green curve with $\nu_{\mathrm{in}} = 14$ Hz. The shown rates are mean values averaged over 250 simulation runs with 10 s duration each. Error bars are standard error of mean. As the membrane of the neuron stimulated with 14 Hz reacts faster on input variations, the test stimulus has a weaker impact on the output rate. Thus $f_{\mathrm{out}} = \nu_{\mathrm{out}}^{\mathrm{stim}} - \nu_{\mathrm{out}}^{\mathrm{nostim}}$ is smaller over the whole T_{ISI} range compared to the neuron stimulated with only $\nu_{\mathrm{in}} = 2\,\mathrm{Hz}$. Furthermore, the test stimulus packages have to be more dense in order to significantly increase the spike probability. This is reflected by a decrease of τ_{res} for larger input rates, i.e. larger input conductances. The simulation was done with the basic parameter set given above.

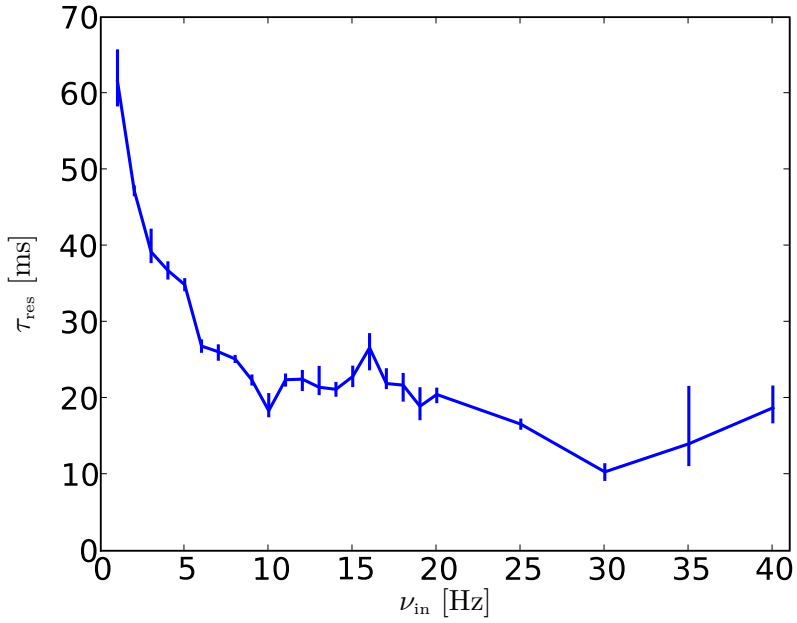


Figure 2.6: Simulation of τ_{res} vs. ν_{in} for a neuron with $\tau_{\mathrm{syn}} = 30$ ms and $g_{\mathrm{l}} = 2\,\mathrm{nS}$. The y-axis represents the temporal resolution capability of the neuron membrane, the x-axis shows the frequency of the Poisson background stimulation, representing the input conductance. With increasing background activity the temporal resolution capability increases. Error bars for τ_{res} are estimated according to the descriptions given in figure 2.4.

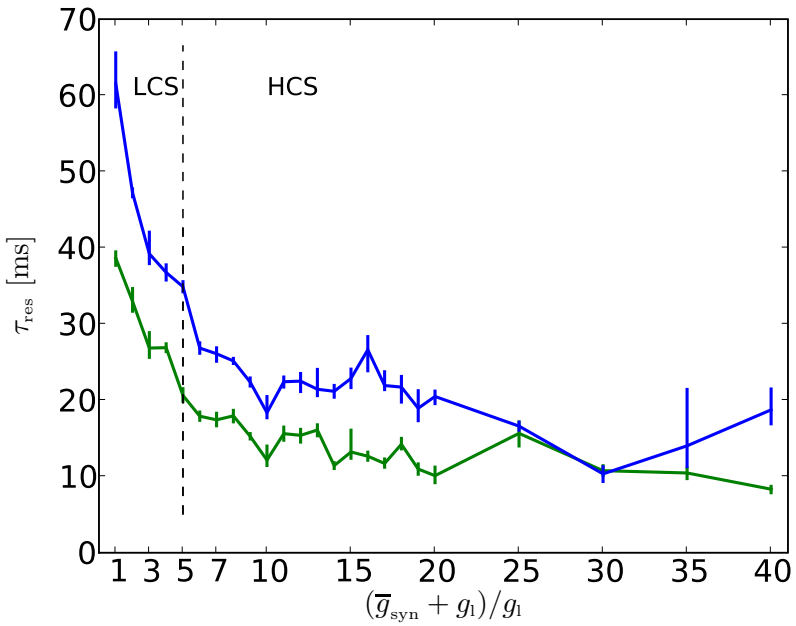


Figure 2.7: NEST Simulation: Separation of conductance regimes with the help of the temporal resolution capability τ_{res} versus the total membrane conductance divided by the pure leakage conductance. Blue curve shows the simulation results from a neuron with $\tau_{\mathrm{syn}} = 30$ ms, green curve with $\tau_{\mathrm{syn}} = 20$ ms, leakage conductance for both is $g_{\mathrm{l}} = 2\,\mathrm{nS}$. The dashed line predicts the separation between low- (LCS) and high-conductance states (HCS) as suggested in [*Brette and Gerstner*, 2005].

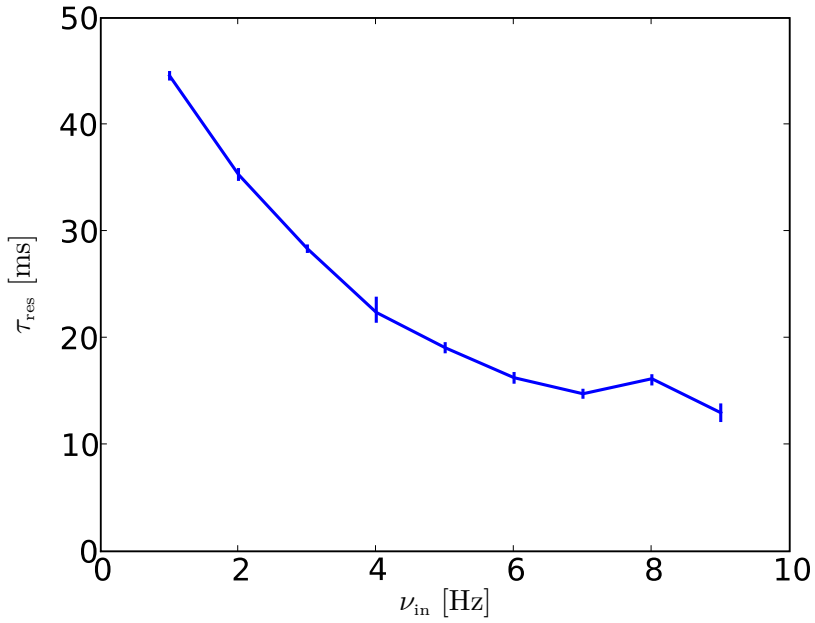


Figure 2.8: Hardware: The time resolution capability of a hardware neuron shows a similar behavior as the neuron used in simulations.

2.2 Synaptic Time Constants

If plasticity mechanisms are ignored, the utilized neuron model contains seven parameters: Three reversal potentials E_E, E_I, E_l, the leakage conductance g_l, the membrane capacitance C_m and two parameters defining a synapse: g_{syn} and τ_{syn}. While the former can be converted rather reliably between biological values and hardware parameters, a proper mapping of g_{syn} and τ_{syn} has not yet been investigated systematically.

The synaptic time constant, especially that of inhibitory synapses, is an important parameter that can have a strong impact on the single neuron and network dynamics [*Amemori and Ishii*, 2000]. For example, in neurons and networks of neurons showing oscillatory activity, the dynamics, in particular the global oscillation frequency, strongly depends on the decay time of the inhibitory post-synaptic currents [*Chow*, 1998; *Hansel et al.*, 1995; *Terman and Wang*, 1995; *Vreeswijk*, 1996]. Furthermore, slow inhibitory synapses can lead to a synchronization of the network activity, which can not be achieved by increasing the amount of inhibition in a different way [*Kumar et al.*, 2008].

The measured ranges of synaptic time constants are an essential information for the cross-inhibition setups investiaged in the following section.

2.2.1 Setup

In the utilized hardware device, synapses are arranged in a 2-dimensional array. Each row of that array represents one axon[1] and is operated by one synapse driver. One column consists the synapses that are connected to one post-synaptic neuron. One synapse driver is either assigned to an external input or a neuron which feeds its input back into the network. When a spike occurs, the synapse driver generates a voltage signal consisting of a steep rising ramp (set via *drvirise*) going up to a maximum (determined by parameter *drviout*) and followed by a falling ramp (adjusted via *drvifall*). In the synapse circuit, this voltage signal is transformed into a current. As the rise time is very small, the current increases nearly instantaneously and then decreases exponentially. The decay time constant τ_{syn} of this exponential current decrease is controlled by the parameter *drvifall*. The influence of *drvifall* on the synaptic time constant is investigated in this section.

In order to measure the synaptic time constant, several aspects have to be considered. As it is not possible to access the current time course on a synapse directly, the desired quantity has to be deduced from the membrane potential. The membrane potential can be easily accessed via an oscilloscope.

As can be seen from the differential equation 1.3 describing the membrane dynamics and derived in detail in [*Shelley et al.*, 2002], the membrane potential follows the time course of input conductances instantaneously, when the total membrane conductance

[1] If a neuron projects onto neurons of both blocks, the axon is represented by two rows – one on each side.

g_{T} is large enough, i.e. the effective membrane time constant $\tau_{\mathrm{m}}(t) = C_{\mathrm{m}}/g_{\mathrm{T}}$ is small enough. This is the case when the neuron is in a so-called high-conductance state due to strong synaptic input, see sections 2.1.1, 1.4 and [*Shelley et al.*, 2002; *Destexhe et al.*, 2003]. Thus, one has to assure that the total synaptic conductance is large enough to be able to deduce the synaptic conductance course from the membrane potential. In the previous section, a high-conductance state test has been introduced for exactly this purpose.

In the following, spike-triggered averaging (see section 1.6) is used to deduce τ_{syn} from the membrane potential $V(t)$. The influence of *drvifall* on the synaptic time constant is studied with only one synapse driver at the same time, i.e. trigger signals are sent by only one synapse driver. The recorded neuron is connected to 120 excitatory and 40 inhibitory inputs which stimulate the neuron with Poisson spike trains of $\nu_{\mathrm{in}} = 5\,\mathrm{Hz}$ for a duration of 100 s. This synaptic background for the STA measurements was chosen here is high enough in terms of the high-conductance state conditions determined in the previous section, which was conducted with approximately the same number of inputs.

A least squares fit of an exponentially decaying function to the mean voltage trace acquired via spike-triggered averaging yields the desired quantity τ_{syn} [*Barlow*, 1989]. Figure 2.9 shows one example of a hardware system PSP extracted in that way.

This procedure is systematically applied for a range of *drvifall* between 0 μA and 2.0 μA. As the integrated current decreases with increasing *drvifall*, the impact of a spike on the recorded neuron decreases as well. In order to keep the mean membrane potential and the total synaptic conductance on their specified working points, the amplitude of the current course, which is controlled by *drviout*, has to be adjusted according to *drvifall*. For *drvifall* values smaller than $0.15\,\mu$A a *drviout* of 0.05 μA was chosen to limit the impact of each spike. For higher *drvifall* values, *drviout* was set to 0.1 μA.

2.2.2 Results

Figure 2.10 shows the measured correlation between *drvifall* and τ_{syn} for five different synapse drivers on one chip. The data shows large synaptic time constants for very small values of *drvifall* and decreasing τ_{syn} for increasing *drvifall*, as expected. However, it is worth noting that an effect of *drvifall* on τ_{syn} can only be observed for values of *drvifall* smaller than 0.02 μA. Larger *drvifall* values do not decrease the mean of τ_{syn} to below 25 ms anymore.

The developing of τ_{syn} for very small *drvifall* is remarkable: τ_{syn} does not decrease continuously, but rather decreases in stages. This is an effect of the limited resolution of the DAC[2] creating the programmed *drvifall* value. The utilized DAC has a resolution of 10 bit and can create values in the range from 0 μA to 2.5 μA. That is, the DAC creates the desired *drvifall* values with a precision of 2.5 μA / 1024 bit = 0.0024 μA /

[2] Digital Analog Converter

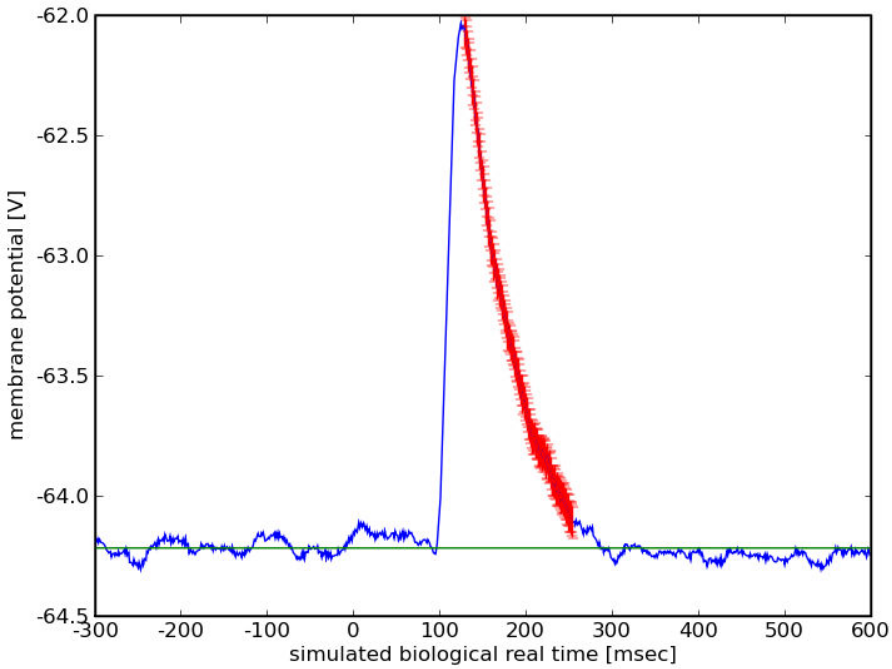


Figure 2.9: Excitatory PSP gained by spike-triggered averaging at $drvifall = 0.004$ μA. Hardware voltage and time scale are transformed to biological scale. The shown PSP is the mean PSP averaged over ca. 2500 samples. The error of each data point i is estimated by: $\sigma_i = \sqrt{(\overline{y_i^2} - \overline{y_i}^2)/(n-1)}$, whereas n is the number of recorded samples and $\overline{y_i}^2$ ($\overline{y_i^2}$) represents the squared average value (the average square value, respectively).

bit. Consequently, pairs of *drvifall* with differences smaller than that do not necessarily show a difference in the resulting τ_{syn}.

A comparison of different synapse drivers yields, that for small *drvifall* values τ_{syn} shows variations between ca. 55 ms and 140 ms in biological time-scale. For $drvifall = 0.025$ μA, the variance of τ_{syn} is minimal and increases for larger *drvifall*. Thus, this parameter should be chosen to guarantee the highest homogeneity for an uncalibrated chip.

The range of time constants τ_{syn}, which can be achieved by a synapse driver, reveals only a minor dependence on the neuron the EPSPs were excited on. Figure 2.11 shows the measurement of two synapse drivers, carried out on two neurons. Obviously, the curve progression is a characteristic of the synapse driver – not of the neuron which was used for the measurement.

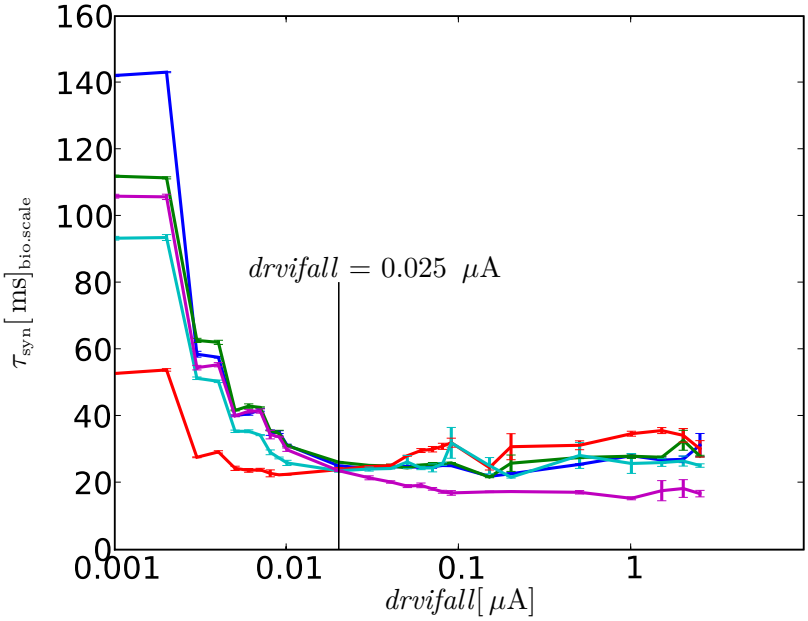


Figure 2.10: τ_{syn} in dependence of *drvifall*. Plotted are synaptic time constants of five different synapse drivers estimated through STA. Each data point was measured three times, errors are standard error mean. Noticeable errors occur for *drvifall* greater than 0.1 μA. The incremental developing of τ_{syn} values at small values of *drvifall* (step size = 0.001 μA) is caused by the limited resolution of the DAC generating the desired *drvifall* values.

Difficulties in measuring τ_{syn}

One problem observed during preparatory investigations of the test setup is that parameter values (e.g. *drvifall*) of different synapse drivers influence each other, which is not yet understood and requires further investigation. A first phenomenological study will be presented in the following.

Figures 2.12a, 2.12b, 2.13a and 2.13b show screenshots from the oscilloscope recording the membrane potential of one neuron. The effect of mutual parameter influence was observed in a setup where only one synapse driver stimulated the recorded neuron, while all other connected synapse drivers (about 100 in total) did not receive any signals. The first figure 2.12a shows, that the *drvifall* value assigned to other synapse drivers strongly affects the shape of the PSP caused by the one observed synapse driver.

A comparison of the two figures in 2.13 shows, that the exact position of the synapse drivers having the same drvifall value does not have an effect on the shape of an EPSP. Concluding, there seems to be no direct electrical interference between synapse drivers due to a close spatial arrangement, but the configuration of an entire synapse driver ar-

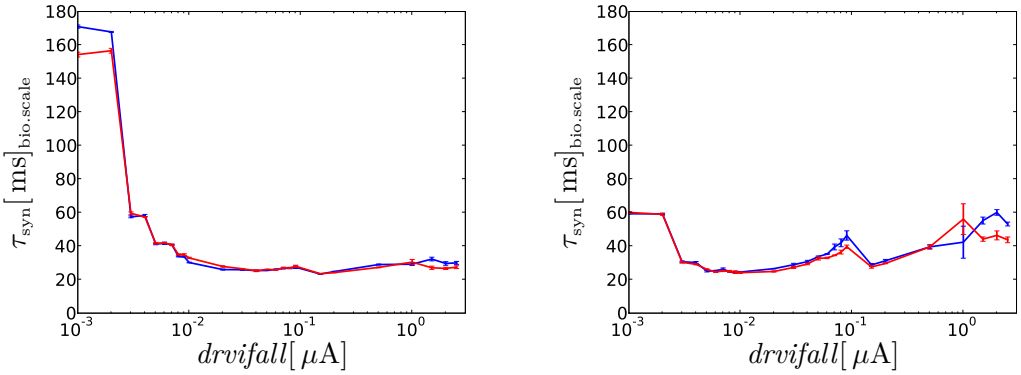


Figure 2.11: Comparison measurement of the synaptic time constant τ_{syn} carried out on different neurons. While different synapse drivers (left and right plot) reveal significantly different curves, the influence of the utilized neuron (red and blue line) is of almost no relevance.

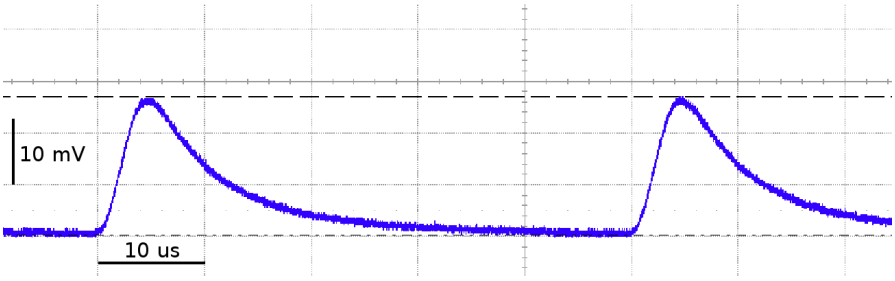
ray reveals a strong interference. It is not clear from the shown figures, which parameter of the observed driver is influenced by others, i.e. whether *drviout* controlling the PSP amplitude or *drvifall* controlling the decay. The typical hardware synapse dynamics significantly changed during the course of working for this thesis, after an error concerning the synaptic resting voltage *VREST* was discovered in November 2008. Therefore, no deeper investigation of the above issue was possible during this thesis. A systematic investigation is strongly recommended.

In order to minimize the undesired mutual influence of parameter values assigned to different synapse drivers, the influence of the number of synapse drivers having the same value as the observed driver is analyzed. Figure 2.14 shows the shape of an EPSP for setups with varying numbers of synapse drivers having the same *drvifall* value. When no other synapse driver has the same value as the observed synapse driver, the EPSP is strongly distorted. With increasing number of synapse drivers having the same value, the shape of the EPSP seems to converge to a 'real' shape. As a consequence of figure 2.14, all previously presented measurements of τ_{syn} in dependence of *drvifall* were carried out with 16 other synapse drivers having the same *drvifall* value as the observed one.

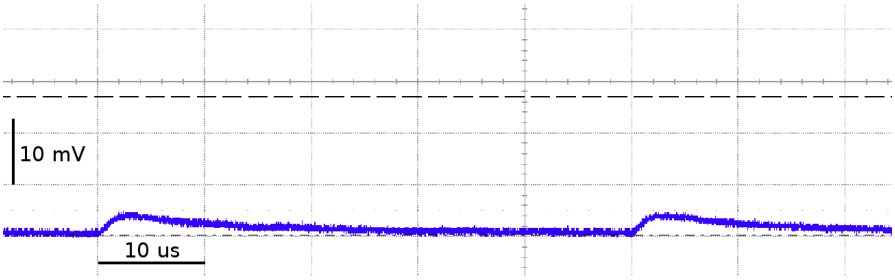
2.2.3 Synaptic Time Constants: Discussion and Conclusion

The presented measurements show, that the synaptic time constant in hardware can be adjusted only in a coarse way. The maximum value for τ_{syn} is dependent on the individual synapse driver and ranges from about 55 ms up to 140 ms. Due to the limited resolution of the DAC generating the desired current, *drvifall* can be set in units of $0.0024\,\mu$A.

For *drvifall* values less than $0.02\,\mu$A, a monotonous correlation is observed as expected. For larger values, an undesired effect concerning the synapse driver reference



(a) Screenshot from the oscilloscope showing EPSPs averaged over 50 runs triggered by **one synapse driver with drvifall = 0.01** μA. 20 **non-adjacent synapse drivers have drvifall = 0.0** μA. Only the synapse driver with drvifall = 0.01 μA sends currents to the recorded neuron, the 20 non-adjacent synapse drivers do not receive any spikes. About 100 synapse drivers are connected without input spikes and have drvifall = 0.15 μA.



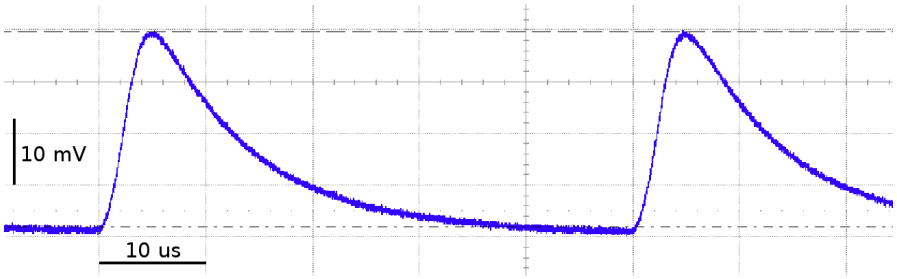
(b) Same setup as above, only difference is that the 20 **non-adjacent synapse drivers** have the same value as the active one, i.e. all **have *drvifall* = 0.01** μA, which leads to a decrease of the PSP-integral. Apparently, the drvifall value of other synapse drivers strongly affects the efficacy of the observed driver.

Figure 2.12: These two figures show the strong effect the *drvifall* value of *other* synapse drivers has on the EPSP shape of a single one.

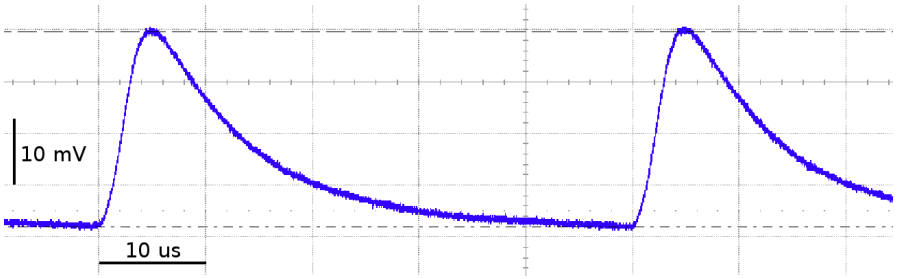
voltages occurs, which was just recently detected and, for the work in this section, made completely new considerations necessary. Too large *drvifall* values increase the synaptic resting voltage *VREST*, which immediately affects the voltage ramp generating the decaying current, which was investigated in this section, but due to the recency of the hardware issue detection, suggests deeper studies. This effect causes a permanent input synaptic conductance and, hence, affects the resting potential V_{rest} of any connected neuron.

The minimum τ_{syn} was determined to be between 30 ms and 20 ms for individual synapse drivers. Since the homogeneity between different synapse drivers is maximum at a *drvifall* value of 0.025 μA and the respective τ_{syn} is close to the minimum, this value seems to be optimal for the operation of the hardware system.

Unfortunately, the synaptic time constants turned out to be not arbitrarily adjustable.



(a) Again, EPSPs averaged over 50 runs triggered by **one synapse driver with *drvifall* = 0.0** μA. 20 **non-adjacent synapse drivers have the same *drvifall* = 0.0** μA. About 100 synapse drivers are connected without input spikes and have *drvifall* = 0.15 μA. Only one sends currents to the recorded neuron. The 20 non-adjacent synapse drivers do not receive any spikes.



(b) Same setup as above, only difference is the **position of the 20 synapse drivers** having the same *drvifall* value as the observed one. In this setup they are **directly adjacent**.

Figure 2.13: Changing the subset of synapse drivers, which are configured with *drvifall* = 0.0 μA, does not influence the observed synapse driver.

Particularly, synaptic time constants smaller than $\tau_{\text{syn}} = 20$ ms do not seem to be possible. Compared to parameter choices as used in typical comparable models [*Kumar et al.*, 2008; *Sussillo et al.*, 2007; *Shelley et al.*, 2002], this is a large value. This is especially true compared to the membrane time constants of about 5 ms to 10 ms inherent to VLSI neurons of the current chip.

Still, for the purpose of cross-inhibition and winner-take-all architectures, which - as will be shown in the following chapter - need large synaptic time constants, the results presented in this section are not necessarily negative. But it remains unclear how problematic the fact that individual synaptic time constants are adjustable only in a rather coarse and unreliable way will turn out for the experiments aimed at.

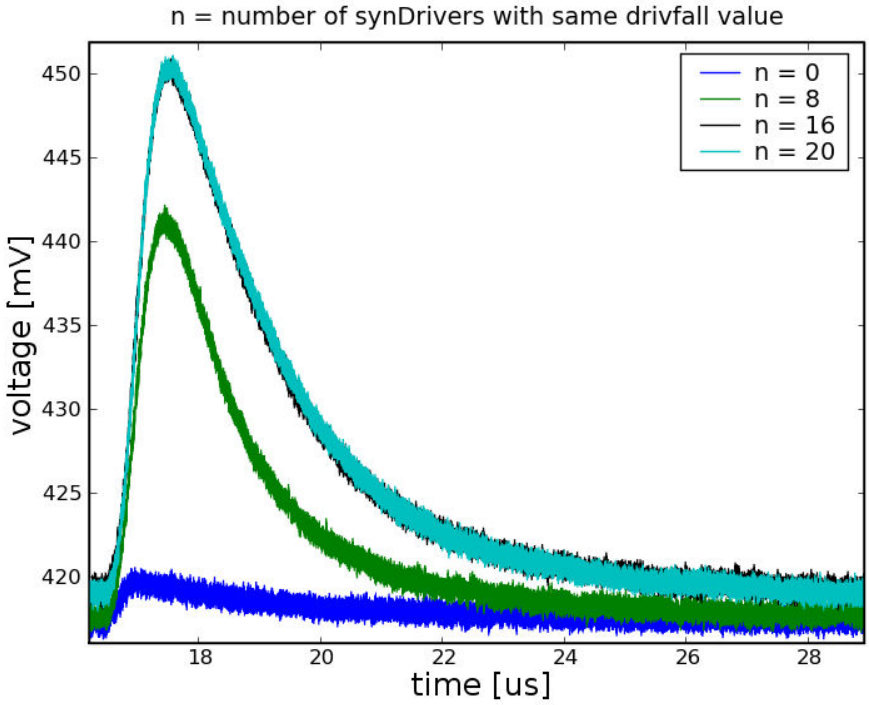


Figure 2.14: Effect of the number of synapse drivers, that have the same value as the observed synapse driver, on an EPSP. For 16 or more synapse drivers, that have the same *drvifall* value, no impact on the observed synapse driver can be seen. Thus, spike-triggered averaging was done with assigning the same *drvifall* value to 16 other synapse drivers. Voltage and time as seen on the oscilloscope.

2.3 Cross Inhibition

Inhibition plays an essential role in the dynamics of neural networks. Inhibitory signals counteract excitatory forces, can interrupt the information flow or restrict activity patterns in space and time and can impose spatial and temporal selectivity [*Buzsaki*, 1984, 1995; *Buzsaki et al.*, 1996; *Watts J*, 2005]. Hence, inhibitory elements are crucial in information transmission, learning processes and memory formation.

One function of strong mutual inhibition can be used to create a selection mechanism, which implements a winner-take-all architecture, see section 2.4, [*Häfliger*, 2007]. In the following, cross inhibition means that all neurons of a network are connected with each other via strong inhibitory synapses. Figure 2.15 shows a schematic of a cross inhibition setup with 4 neurons.

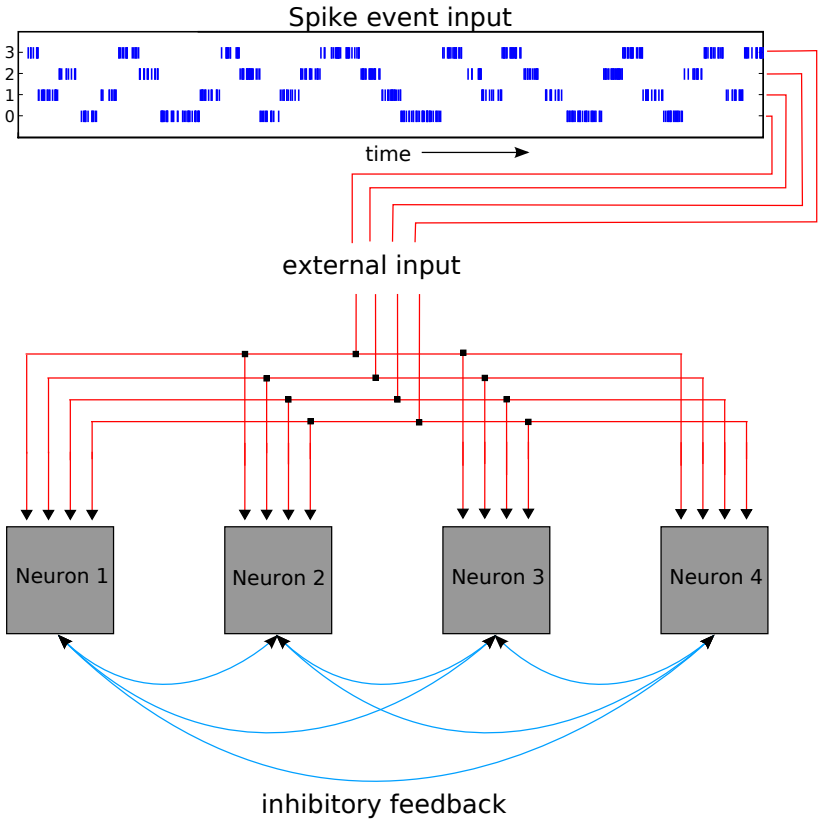


Figure 2.15: Schematic of a cross inhibition setup with 4 neurons and 4 inputs. Each neuron receives each external input plus feedback from each neuron except of itself. All neurons receive the same temporal information, but as external input weights are randomly distributed, incoming spikes have different impact on each neuron.

In this section, the influence of various parameters on networks implementing cross inhibition is investigated. The experimental setup and important parameters are introduced in subsection 2.3.1. In order to describe the behavior quantitatively, a measure for the performance is defined in subsection 2.3.2. Utilizing this performance measure, the influence of different network parameters is analyzed with the help of software simulations in subsection 2.3.3. In section 2.3.5, conclusions from these simulations are drawn with respect to winner-take-all architectures that are investigated in the course of this thesis. Furthermore, the feasibility of porting these experiments onto the hardware system is discussed.

2.3.1 Setup

In a cross inhibition setup, the neural network consists of N inhibitory neurons. The output of each neuron is connected to any other neuron within the network with probability p_{feedback}. The weights g_{feedback} of connections between neurons are constant over time, just like the weights for external stimulation, so no learning rule is applied in this setup. Connections with itself are not possible. Each neuron is connected to the same n_{in} external inputs, which stimulate the network with Poisson type spike trains. The weights of synapses are normally distributed (see figure 2.16), so that the external inputs have different impact on each neuron. If all synaptic weights g_{exc} had exactly the same value, all neurons would spike at exactly the same time and a single winner neuron could not be determined. As the cross inhibition setup investigated here is intended to implement a winner-take-all functionality, it is indispensable that external synaptic weights have non-uniform values. The optimum width of the Gaussian distribution from which the external weights are drawn is investigated for one setup in section 2.3.3.

In the hardware system, all components are subject to process variations, which also leads to variations in the synaptic efficacies. Supposed that these variations are significantly smaller than the optimum of variations identified by the simulations, an additional weight modification would be necessary. Since hardware weights can only be set with a resolution of 4 bit, the influence of quantized weights is investigated.

For a first study, all feedback weights are uniform, i.e. all inhibitory connections between neurons have the same value. In a second series of experiments, the influence of normally distributed feedback weights is investigated, since in the hardware system all components are subject to fluctuations.

The parameters defining the cross inhibition setup are:

- N: number of neurons in the network
- g_{exc}: weights for excitatory synapses (external stimulation)
- g_{inh}: weights for inhibitory synapses (external stimulation)
- $\sigma_g/\overline{g}$: relative weight variation parameter for external synapses, i.e. width of the Gaussian distribution relative to the mean value of the distribution ($\overline{g} \neq 0$)
- $n_{\text{exc}}, n_{\text{inh}}$: number of external inputs without feedback connections $n_{\text{in}} = n_{\text{exc}} + n_{\text{inh}}$
- ν_{in}: frequency of external stimulation
- g_{feedback}: weights for inhibitory feedback synapses
- p_{feedback}: probability with which each neuron is connected to another neuron
- τ_{syn}: synaptic time constant, i.e. decay constant for the conductance time course

For the whole set of neuron parameters see appendix A.1.

As it is unclear which set of parameters leads to the best performance, runs with different parameter sets have to be performed for a single network size.

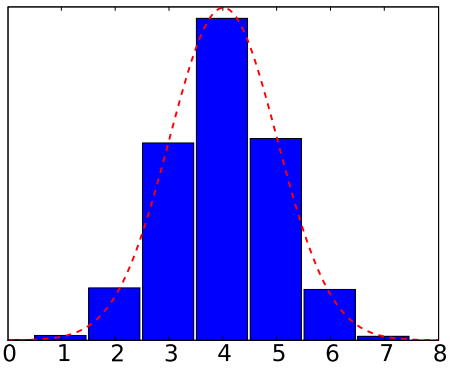


Figure 2.16: In simulations, synaptic weights for external connections are normally distributed (red curve). This is necessary in order to have a different impact of an external stimulus on individual neurons. Shown is a Gaussian distribution with $\mu = 4, \sigma = 1$ (arbitrary units). In the case, that the hardware intrinsic diversity of weights is not enough to guarantee sufficient cross inhibition performance, an additional modification has to be applied to synapses. As synapse weights are quantized in the hardware system, weights would be drawn from the quantized distribution (blue histogram). Presumably, the intrinsic diversity of hardware synapse weights is large enough to enable different impact of a stimulus on individual neurons. The extent of diversity is expressed by the fraction $\sigma/\mu = \sigma_g/\overline{g}$. The influence of $\sigma_g/\overline{g}$ on the cross inhibition performance is studied in simulations in subsection 2.3.3.

2.3.2 Performance Measure for Cross Inhibition

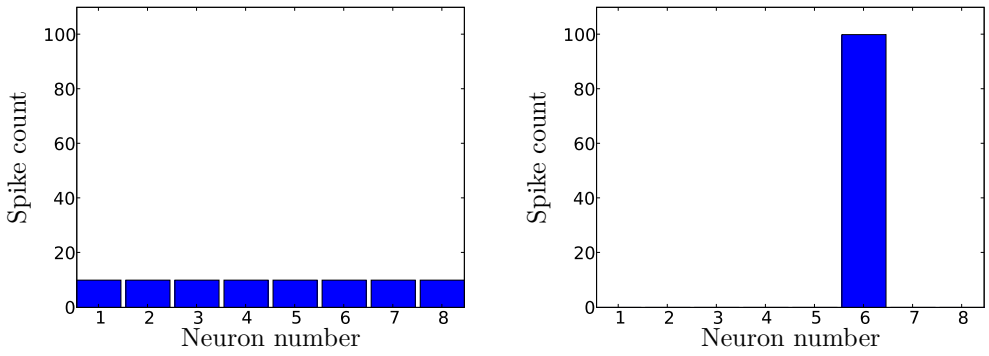
In order to quantify the performance of a cross inhibition setup that is intended to implement a WTA architecture, a quality measure is defined as follows:

$$Q_{\text{ci}} = \frac{2s^* - S}{S} \tag{2.3}$$

The quantities used to define this quality measure Q_{ci} are:

- $S = \sum_{i=1}^{N} s_i$: total number of output spikes fired by the network, s_i: output spike count of neuron i
- $s^* = \max_{i=1..N} \{s_i\}$: number of spikes fired by the most active neuron

Q_{ci} is positive if one neuron spikes more often than all other neurons together: $Q_{\text{ci}} > 0$, if $s^* > S - s^*$. The desired behavior of a network implementing cross inhibition is shown in figure 2.17b and 2.18b. In fig. 2.17b Q_{ci}, is equal to 1, as $S = s^*$, i.e. only one neuron spikes and inhibits all other neurons from spiking. If the spike output is distributed



(a) Evenly distributed spikes **(b)** Ideally distributed spikes

Figure 2.17: Exemplary histograms showing two possible spike count distributions with the same total number of output spikes $S = 100$. An even distribution leads to $Q_{\text{ci}} = -0.75$, an ideal to distribution $Q_{\text{ci}} = 1$

uniformly over all neurons, i.e. $s^* = S/N$, which represents the worst functionality of a cross inhibition setup that is aimed to implement a winner-take-architecture, $Q_{\text{ci}} = \frac{2}{N} - 1$ Thus, in this case Q_{ci} is close to -1 for large networks. If the network does not spike at all, Q_{ci} is set to -1 by definition. Hence, Q_{ci} maps the spike output distribution into the interval $[-1, 1]$. Since it clearly assesses the dominance of one neuron, Q_{ci} is an adequate quantity to represent the effect of cross inhibition on the network behavior.

However, the presented measure Q_{ci} does only reflect the *relation* between the maximum spike count s^* and the total spike output S. Q_{ci} does not reflect any information about the *total number* of spikes fired by the network. Thus, additional information about the total spike count has to be taken into account in order to judge whether the network shows the desired behavior, i.e. if the target output rate is reached. For example: When only one spike is fired, $Q_{\text{ci}} = 1$. Provided the desired output rate is higher, the network does not show the desired behavior, although Q_{ci} equals one. This is especially important for WTA architectures, where the desired learning effect is dependent on the output spike rate.

The importance of total output rates during in the quality assessment is strongly dependent on the specific problem.

A more general definition for a cross inhibition performance measure is:

$$Q_{\text{ci}} = \frac{\alpha \cdot s^* - S}{S} \cdot \frac{1}{\alpha - 1} \tag{2.4}$$

whereas $\alpha > 1$.

α stands for the fraction the maximum spike count s^* has to exceed of the total spike count S so that Q_{ci} is positive:

$$Q_{\text{ci}} > 0 \quad \Longleftrightarrow \quad s^* > \frac{S}{\alpha} \tag{2.5}$$

The factor of $1/(\alpha - 1)$ in equation 2.4 guarantees that $Q_{\text{ci}} \leq 1$. The smaller α is, the more strict the assessment of the spike output is with respect to a perfect winner-take-all output. The larger α is, the smaller s^* has to be for a $Q_{\text{ci}} > 0$: e.g. $\alpha = 5$ requires that the most active neuron fires more that 20 % of the total spike output in order that $Q_{\text{ci}} > 0$.

In the following the utilized Q_{ci} is defined as in equation 2.3, i.e. $\alpha = 2$.

2.3.3 Simulation Parameters and Results

In order to find parameters that enable a good performance of a WTA architecture, networks with different parameters were simulated using the software simulator NEST [*Morrison et al.*, 2007, 2005]. More information about the utilized software framework can be found in section 1.3

In the design of the presented simulations, the feasibility of porting the setup to the hardware system was considered. That is, hardware constraints as e.g. the limited number of recordable neurons, the limited number of inputs for each neuron, input bandwidth limitations and quantization of synaptic weights were taken into account when defining the simulation parameters.

If not given explicitly, parameter values for simulations are the following:

n	n_{exc}	g_{exc}	g_{inh}	$\sigma_g/\overline{g}$	p_{feedback}
16	48	$4 \cdot 10^{-4}\,\mu$S	$64 \cdot 10^{-4}\,\mu$S	0.1	1.0

Table 2.1: Cross inhibition simulation parameters

Influence of ν_{in} and τ_{syn}

In order to achieve a good cross inhibition performance, a certain minimum stimulation is required, since inhibition needs activity. Two parameters that control the stimulation efficacy are the synaptic time constant τ_{syn} and the frequency ν_{in} of the Poisson type input spike trains. In the utilized model, the synaptic time constant is the decay constant of the conductance time course resulting from incoming action potentials – for excitatory as well as for inhibitory connections. For efficient inhibition leading to a single winner neuron, it is necessary that PSPs[3] overlap: Inhibitory PSPs from other neurons overlap

[3] PSP: post-synaptic potential

with excitatory ones from the external stimulus and thus impede spiking. This overlap of PSPs can be realized by a high input frequency or wide PSPs. As the input rate is limited in the hardware system, it is necessary to find an adequate value for τ_{syn} which allows the superposition of PSPs.

Figure 2.19 shows simulations of networks with $n = 16$ neurons. On the x-axis, the synaptic time constant τ_{syn} is varied between 10 ms and 60 ms. This range was chosen with respect to the range of τ_{syn} in the hardware system, see section 2.2. As shown there, the lower bound for the utilized chip was found to be around 25 ms, but for future revisions of the chip, faster synapses are planned.

On the y-axis, the frequency of the Poisson spike trains is varied between 4 Hz and 20 Hz, also reflecting hardware possibilities. Each neuron receives Poisson type spike trains from $n_{\text{exc}} = 48$ excitatory synapses. All applied parameters are listed in table 2.1.

Figure 2.19a shows the total output rate $\nu_{\text{out}}^{\text{tot}} = \sum_{i=1}^{16} \nu_{\text{out,i}}$ of the network under different values from ν_{in} and τ_{syn}, which ranges from 0 Hz to (244.3 ± 0.5) Hz. Figure 2.19b shows the performance measure Q_{ci} defined above for the same range of values. The lowest value with $\nu_{\text{out}} > 0.1$ Hz is $Q_{\text{ci}} = -0.33 \pm 0.03$ for $\tau_{\text{syn}} = 10\,\text{ms}$ and $\nu_{\text{in}} = 18\,\text{Hz}$. The highest one is $Q_{\text{ci}} = 0.998 \pm 0.001$ for $\tau_{\text{syn}} = 60\,\text{ms}$ and $\nu_{\text{in}} = 6\,\text{Hz}$. The given values are the mean values averaged over 50 runs each with 5 s duration; errors are standard errors of mean, see figure 2.22b for numerical values.

The figure shows, that for a certain input frequency ν_{in} the synaptic time constant τ_{syn} has to be chosen high enough to enable a large value of Q_{ci}. For example for an input rate of $\nu_{\text{in}} = 10\,\text{Hz}$, a synaptic time constant of at least $\tau_{\text{syn}} = 30\,\text{ms}$ is preferable (see fig. 2.19b, $\tau_{\text{syn}} \geq 30\,\text{ms}$).

Figure 2.19a shows the total output rate $\nu_{\text{out}}^{\text{tot}} = \sum_{i=1}^{16} \nu_{\text{out,i}}$ of the network under different values from ν_{in} and τ_{syn}, which ranges from 0 Hz to (244.3 ± 0.5) Hz. Figure 2.19b shows the performance measure Q_{ci} defined above for the same range of values. The lowest value with $\nu_{\text{out}} > 0.1$ Hz is $Q_{\text{ci}} = -0.33 \pm 0.03$ for $\tau_{\text{syn}} = 10\,\text{ms}$ and $\nu_{\text{in}} = 18\,\text{Hz}$. The highest one is $Q_{\text{ci}} = 0.998 \pm 0.001$ for $\tau_{\text{syn}} = 60\,\text{ms}$ and $\nu_{\text{in}} = 6\,\text{Hz}$. The given values are the mean values averaged over 50 runs each with 5 s duration; errors are standard errors of mean, see figure 2.22b for numerical values.

The figure shows, that for a certain input frequency ν_{in} the synaptic time constant τ_{syn} has to be chosen high enough to enable a large value of Q_{ci}. For example for an input rate of $\nu_{\text{in}} = 10\,\text{Hz}$, a synaptic time constant of at least $\tau_{\text{syn}} = 30\,\text{ms}$ is preferable (see fig. 2.19b, $\tau_{\text{syn}} \geq 30\,\text{ms}$).

An explanation for this is the competition between external excitation and internal inhibition: If the synaptic time constant is large, the effect of an incoming spike on a neuron's membrane potential is strong because more charge is injected into the neuron. As action potentials decrease more slowly with larger τ_{syn}, they interfere with each other with higher probability. Hence, inhibitory spikes from one neuron have stronger impact on others and cross inhibition works more efficiently. Consequently, the performance of this experimental setup is strongly dependent on synaptic time constants.

As already mentioned, the input rate also determines the probability of an overlap of PSPs. An increase of ν_{in} intensifies the competition between excitation and inhibition. This is demonstrated in figure 2.20, which shows the dependency of Q_{ci} on the external stimulation frequency ν_{in} For growing ν_{in}, Q_{ci} first increases (6 Hz $\leq \nu_{\text{in}} \leq$ 12 Hz) and then decreases for $\nu_{\text{in}} > 12\,\text{Hz}$, even though the maximum output rate s^* further increases. This is due to the external excitation, which counteracts the mutual inhibition and for high input rates finally exceeds it.

Influence of ν_{in} and τ_{syn} with constant global output rate $\nu_{\text{out}}^{\text{tot}}$

The previous investigations might suggest that cross inhibition is dependent on the external stimulation and hence on the output rate, see figure 2.20. In order to investigate the influence of τ_{syn} independent from the output rate, a setup was created that adjusts the global output rate by changing the number of excitatory and inhibitory inputs. The number of excitatory and inhibitory inputs were changed between runs with different values for ν_{in} and τ_{syn}, whereas $n_{\text{exc}} \cdot g_{\text{exc}} + n_{\text{inh}} \cdot g_{\text{inh}}$ is kept constant. The target global output was $\nu_{\text{out}}^{\text{tot}} \geq 75$, Hz which was reached for most of the scanned parameter range, except for very weak external stimulation (bottom left in figure 2.21a). The colormap 2.21b shows a decrease of Q_{ci} with increasing external stimulation frequency ν_{in} (y-direction). Furthermore, figure 2.21 verifies the dependency of Q_{ci} on the synaptic time constant τ_{syn}. Consequently, an optimal choice of parameters would be rather long synaptic time constants $\tau_{\text{syn}} \geq 30$ ms and input rates less than 10 Hz.

Influence of weight quantization

The quantization of synaptic weights in the hardware system is a restriction, which does not exist in that way under biological conditions. In order to represent the weight quantization in simulations, synaptic weights were rounded to multiples of $10^{-4}\,\mu$S, which is a realistic value after first hardware calibration attempts. The influence of quantized weights was tested by subtracting Q_{ci}(without weight quantization) from Q_{ci}(with weight quantization) for the parameter range given above. Simulation parameters were those listed in table 2.1. Figure 2.22 shows that the difference $\delta_{\text{Q,ci}}$ is in the order of magnitude of the standard error of mean. Hence, for the chosen setup, the quantization of weight is no restriction that strongly affects the performance of cross inhibition.

Influence of weight variation $\sigma_g/\overline{g}$

As every VLSI device is subject to process variations, the influence of variations in synaptic weights deserves study. In order to simulate hardware intrinsic variations in synaptic weights, the weights for the simulations were drawn from a Gaussian distribution with standard deviation σ_{g}. Figure 2.16 exemplary shows a Gaussian distribution with a mean value equal to $g_{exc} = 4$ and $\sigma_{\text{g}} = 1$ (in arbitrary units)

The effect of σ_{weights} on Q_{ci} for fixed values of the input rate $\nu_{\text{in}} = 12$ Hz and $\tau_{\text{syn}} = 30$ ms is shown in figure 2.23 with $n_{\text{exc}} = 48, g_{\text{exc}} = 4 \cdot 10^{-4}\,\mu$S and in figure 2.24 with $n_{\text{exc}} = 96, g_{\text{exc}} = 2 \cdot 10^{-4}\,\mu$S. Other simulation parameters are as given above (see table 2.1), except that $\sigma_g/\overline{g}$ is varied between 0 and 2.0. The blue curve shows Q_{ci} for a setup where only the external weights are normally distributed. The green curve shows Q_{ci} for a setup where additionally the inhibitory feedback weights are normally distributed, which is a more realistic scenario in hardware, where all synaptic weights are affected by fluctuations. The ratio between standard deviation and mean value $\sigma_g/\overline{g}$ also holds for the feedback weights, which have a much higher weight.

The plots (2.23 and 2.24) show a non-monotonous relation between the cross inhibition performance measure Q_{ci} and $\sigma_g/\overline{g}$. If σ_{weights} equals zero, all neurons get exactly the same stimulation. Thus all neurons fire with exactly the same rate: $s^* = S/N$ and $Q_{\text{ci}} = 2/16 - 1 = -0.875$. When the weight variation parameter $\sigma_g/\overline{g}$ is increased, Q_{ci} reaches a maximum and then decreases for increasing variation $\sigma_g/\overline{g}$. A comparison between the two setups – uniform feedback weights (fig. 2.23) and non-uniform feedback weights (fig. 2.23) – shows that the setup with non-uniform feedback weights is more sensitive to the weight variation parameter $\sigma_g/\overline{g}$, which is explained in the following.

The fact that cross inhibition works worse when feedback weights are non uniform compared to uniform feedback weights can be explained as follows: On the network scale, variations in the positive direction for excitatory synaptic weights (i.e. stronger synapses), which promotes a winner neuron, can be compensated by variations in the negative direction for inhibitory synaptic weights (i.e. weaker synapses), see figure 2.25b). In other words, it can easily occur that inhibitory connections going away from one strongly stimulated neuron are weaker than the average and that connections going away from other neurons and converging to the strongly stimulated neuron are strong. In that case the strongly stimulated neuron would spike often, but could not inhibit other neurons sufficiently.

Figure 2.25b illustrates such a setup that is unfavorable in terms of a winner-take-all architecture. Neuron 2 is strongly excited by the external inputs but only weakly inhibits neuron 1. The situation is inverse for neuron 1: it has weak connections to the external inputs, but in the case neuron 1 fires a spike, the IPSP[4] has strong impact on neuron 2. So, both are able to fire and hence reduce Q_{ci}. This effect is stronger when $\sigma_g/\overline{g}$ is large, which is confirmed by figure 2.23.

Figure 2.24 shows Q_{ci} against $\sigma_g/\overline{g}$ for a setup with twice the number of excitatory inputs and half the weights compared to figure 2.23, i.e. $n_{\text{exc}} \cdot g_{\text{exc}} = const$ for both setups. A comparison between the two plots yields that a larger number of inputs with smaller weights benefits cross inhibition quality in two ways: The maximum value of Q_{ci} is higher in fig. 2.23a than in fig. 2.24a and the decrease due to unfavorable weight distribution begins at larger $\sigma_g/\overline{g}$. This can be explained by statistical balancing of the

[4] Inhibitory Post-Synaptic Potential

synaptic weights: For small numbers of inputs n_{in}, an unfavorable weight distribution has stronger impact than in the case when n_{in} is large. Thus, a setup with a larger number of inputs and weaker connections is more robust against the negative effects of weight variations.

Influence of the network size

It is a basic question how strong the number of neurons in the network influences the cross inhibition performance. The more neurons the network comprises, the more inhibitory connections exist for each neuron. Thus, it is more difficult for a neuron to prevail. This is reflected by the fact that Q_{ci} decreases with increasing total number of spikes: $Q_{\text{ci}} = 2\frac{s^*}{S} - 1$.

As the total number of spikes increases with the number of neurons, the performance measure is expected to decrease with increasing network size. This is confirmed by figure 2.26. The data shown in figure 2.26 represents mean values over 50 runs with 5s duration each. Simulations were done with $\tau_{\text{syn}} = 30\,\text{ms}$, $\nu_{\text{in}} = 12\,\text{Hz}$ and the parameters given in table 2.1. The figure shows two curves, where Q_{ci} is plotted against the number of neurons in the network. The curve in blue represents a setup with uniform feedback weights. The green curve represents a setup with normally distributed feedback weights and again shows the strong effect originating from disadvantageous weight distribution discussed above (see e.g. figure 2.25b). Every setup was simulated with the same amount of external stimulation, thus the number of inhibition increases with the number of neurons, whereas excitatory inputs remain constant. To facilitate cross inhibition performance in larger networks, external stimulation probably needs to be increased in order to counterbalance the increased number of inhibitory connections. The following studies will concentrate on small network sizes, i.e. $N = 16$ neurons.

2.3.4 Hardware Implementation

The fundamental difference in the underlying platform when comparing software simulations with emulations on the hardware is the homogeneity of the substrate. In software experiments all components of the same type are perfectly identical in behavior and arbitrarily configurable. Hardware components are subject to process variations, and electronic phenomena like noise and temperature dependent effects and can only be configured within a certain range. Thus, equal results cannot be expected. In order to counterbalance inhomogeneities in the electronic circuits, different calibration routines were applied to the utilized chip.

This section concludes the results of a first effort of porting the cross inhibition setup onto the hardware. As membrane capacity, threshold voltage, leakage current, reset pulse strength and efficacies of the connected synapses are individual for each neuron circuit, the first step is to analyze the firing behavior of different neurons. This is done by

stimulating each neuron on the chip with the same number of input spike trains injected through the same synapse drivers. Figure 2.27 shows a histogram of mean spike rates averaged over 10 runs with standard deviation as error bars. Each neuron was stimulated with 48 excitatory and 12 inhibitory Poisson spike trains for 100 s (biological timescale) connected with hardware weights of $g_{\text{exc}} = 4$ bit and $g_{\text{inh}} = 12$ bit. The number of inputs was chosen in correspondence to the simulations discussed above.

The plotted mean spike rates show large variations between neurons. In figure 2.27a, a small difference between the two neuron blocks can be recognized at neuron number 192. Figure 2.27b shows the same data set sorted by the output rate, neglecting neurons that do not spike at all with the given stimulation. Figure 2.27b illustrates that the number of neurons spiking at the same mean rate is very small.

One further problem, that makes it difficult to apply the presented cross inhibition setup on the current revision of hardware system is the limited number of neurons, that are recordable at the same time. In general there are ca. 24 neurons per neuron block recordable at the same time, i.e. ca. 48 neurons per chip. This number is distributed over the chip in 6 groups with ca. 8 adjacent neurons each. As can be seen in the histogram in figure 2.27a, adjacent neurons do in general not show similar spiking behavior. As homogeneous spiking behavior is highly desirable, it is not a promising approach to take the 48 recordable neurons for the cross inhibition setup without checking their output rate.

Thus, a different approach was chosen which makes use of figure 2.27b showing the mean output rates. 16 neurons with similar output rates were selected from the different groups of recordable neurons. The synaptic weights were adjusted so that the mean output rate was as equal as possible. Again all neurons are stimulated with the same Poisson spike trains and no mutual inhibitory connections are drawn. The resulting distribution is shown in in figure 2.28a. Plotted are mean values averaged over 100 runs with 10 s (biological timescale) each.

When cross inhibition is applied and the output of each neuron is fed back to all other neurons via inhibitory synapses, the firing rate decreases (see figure 2.28b). As expected, the neurons that were most active in the setup without cross inhibition, dominate the output also in this setup with cross inhibition. The output spike distribution for each run is shown in figure 2.28c. This figure shows the neuron number on y-axis and the run number on the x-axis. The output spike count is represented by the gray scale ranging from 0 to 35. The plot clearly shows, that the output distribution is very similar for each run. Thus, it is very often the case, that the same neuron dominates the output and inhibits others, which is not desired with respect to the self-organized WTA learning experiment presented in the next section.

Left for further studies, there are several possibilities to improve the cross inhibition performance on the hardware. One possibility is to increase the number of inputs which enables more possibilities for the weight tuning algorithm in order to counterbalance weight variation and make the output rates more uniform. Regarding this point, the

limited number of inputs has to be kept in mind, which limits the number of possibilities. Furthermore, the strength of the synapse drivers responsible for inhibitory feedback need to be calibrated for stronger inhibitory effect. However, this does not counteract the inhomogeneities in neuron spiking behavior, which is mainly due to an individual threshold voltage. This problem will partly be solved in the next revision of the chip, which will reduce undesired correlation between the threshold voltage and the reset pulse strength. Further promising possibilities to counterbalance inhomogeneities might be the usage of populations instead of single neurons as mutually inhibiting units, and possibly within each unit a self-stabilizing architecture based on short-term synaptic plasticity as described in [*Sussillo et al.*, 2007; *Bill*, 2008].

2.3.5 Cross Inhibition: Discussion and Conclusion

In this section, the performance of various cross inhibition setups was studied. A quality measure Q_{ci} was introduced that allows to evaluate the performance quantitatively. Simulation results yield an estimation for a minimal stimulation and ranges for τ_{syn} and ν_{in} that lead to large Q_{ci}.

The influence of quantized weights compared with continuous weights turned out to be negligible, i.e. in the order of magnitude of statistical fluctuations. Thus, no disadvantage for the performance due to weight quantization has to be expected for the hardware system.

On the contrary, variations in synaptic weights do have a strong influence on the quality of cross inhibition and indicate the need for a proper calibration of the hardware system in order not to influence the performance too strong. On the one hand, external synaptic weights must not have the same value, on the other hand Q_{ci} is very sensitive to the weight variation parameter and large values of $\sigma_g/\overline{g}$ can seriously decrease Q_{ci}. This is especially true when additionally to the weights for external connections the weights for feedback connections are subject to variations (see figures 2.25, 2.23). It was shown, that a larger number of stimuli with smaller weights can counterbalance the weight variation for external connections (compare figures 2.23 and 2.24). As the number of inhibitory connections is strictly determined by the number of neurons in the network, it is not possible to counterbalance the variations in feedback weights by increasing the number of connections.

It can be assumed, that the intrinsic variations in the hardware system exceeds the minimum weight variation needed to enable cross inhibition and that an additional modification is not necessary. In the hardware system the following aspects concerning $\sigma_g/\overline{g}$ have to be considered: The strengths of synapse drivers vary by 10 % to 30 % [*Brüderle*, 2008]. Thus, each synapse driver has a different impact on one neuron even after calibration which attempts a uniform impact. Additionally, each neuron has an individual time constant, membrane capacity and spike threshold and is influenced by different reversal potentials due to process variations in the neuron circuit which affects

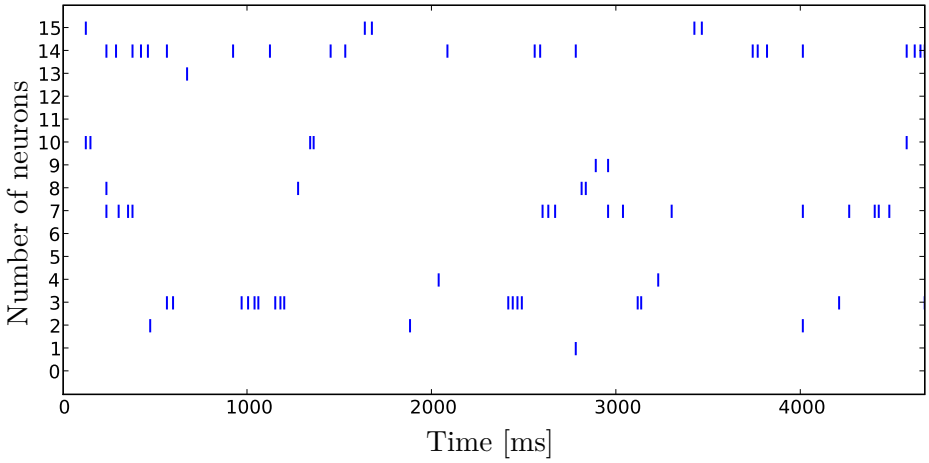
subthreshold behavior and spike probability. Hence, it is not possible to adjust $\sigma_g/\overline{g}$ to arbitrary values. An estimation of $\sigma_g/\overline{g}$ can be found in [*Bill*, 2008, section IV.3] which yields that $\sigma_g/\overline{g}$ is in the order of magnitude of 10 % when an ideal calibration of the PSP integrals is preconditioned.

Furthermore, networks of different sizes were compared. The more neurons the network comprises, the more inhibitory inputs exist for each neuron. As expected, smaller networks perform better than large ones, because in large networks it is more difficult for one neuron to prevail. Thus, an optimal setup would consist of a small number of competing neurons, that share a large number of external inputs with small weights for stimulation. To counterbalance increased inhibition for larger networks, probably stronger excitation in form of larger $\nu_{\text{in}}, n_{\text{in}}$ or g_{exc} is required to improve the cross inhibition performance. The detailed parameters under which cross inhibition works in the desired way for larger networks demand further studies. Another important parameter for the quality of cross inhibition, which was not studied here, is the weight of inhibitory connections g_{feedback}, which has to be chosen sufficiently high to enable effective mutual inhibition.

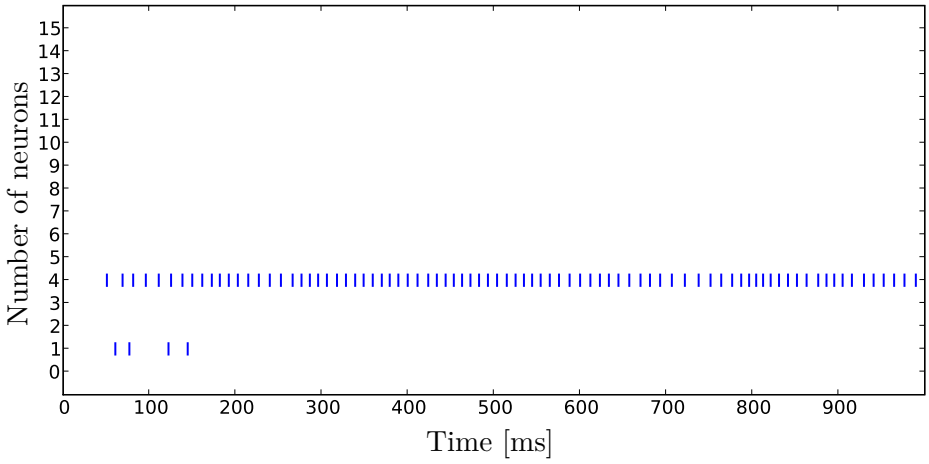
As these quantities $\nu_{\text{in}}, n_{\text{in}}$ and g_{exc} are limited in the hardware system, it is desirable to adjust synaptic time constants in order to achieve the necessary amount of stimulation given to the network.

The most important conclusion to be drawn from this section is that synaptic time constants play a crucial role in this setup with respect to winner-take-all architectures. Too small time constants prevent the superposition of PSP on a neuron's membrane. As this superposition of inhibitory PSPs from feedback synapses and excitatory PSPs from external stimuli is necessary for a good cross inhibition performance, the synaptic time constant is a fundamental parameter that has to be adjusted properly.

As in the PyNN implementation of NEST2 the synaptic time constants are implemented as part of the neuron parameters, the influence of variations in the synaptic time constants (i.e. different τ_{syn} for one neuron) can not be studied in a way that realistically resembles the hardware system and biological realistic setups.

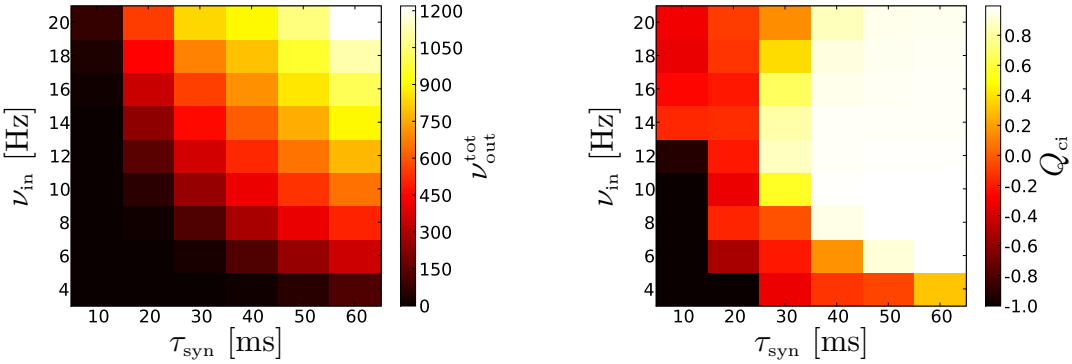


(a) $\tau_{\text{syn}} = 20\text{ms}$, $Q_{\text{ci}} = -0.429$



(b) $\tau_{\text{syn}} = 40\text{ms}$, $Q_{\text{ci}} = 0.982$

Figure 2.18: Rasterplots showing the spike output from a network of 16 neurons implementing cross inhibition with different synaptic time constants, but identical external input. The network of 16 neurons is stimulated with $\nu_{\text{in}} = 12\,\text{Hz}$. In fig. 2.18a, the synaptic time constant is $\tau_{\text{syn}} = 20\,\text{ms}$, which is too small to enable efficient cross inhibition. Fig. 2.18b shows the same setup with larger τ_{syn}. The temporal sequence of input spikes is identical in both setups. The effect of incoming spikes is stronger in figure 2.18b due to larger τ_{syn}. After ca. 150 ms, one neuron continuously dominates the output and inhibits other neurons from firing, which leads to large Q_{ci}.



(a) Total output rate in a network of 16 neurons implementing cross inhibition.

(b) Cross inhibition performance

Figure 2.19: Color map showing the total output rate and cross inhibition performance Q_{ci}, respectively, against synaptic time constant τ_{syn} and input rate ν_{in}. Plotted are the mean values averaged over 50 runs. Simulations parameters are according to table 2.1.

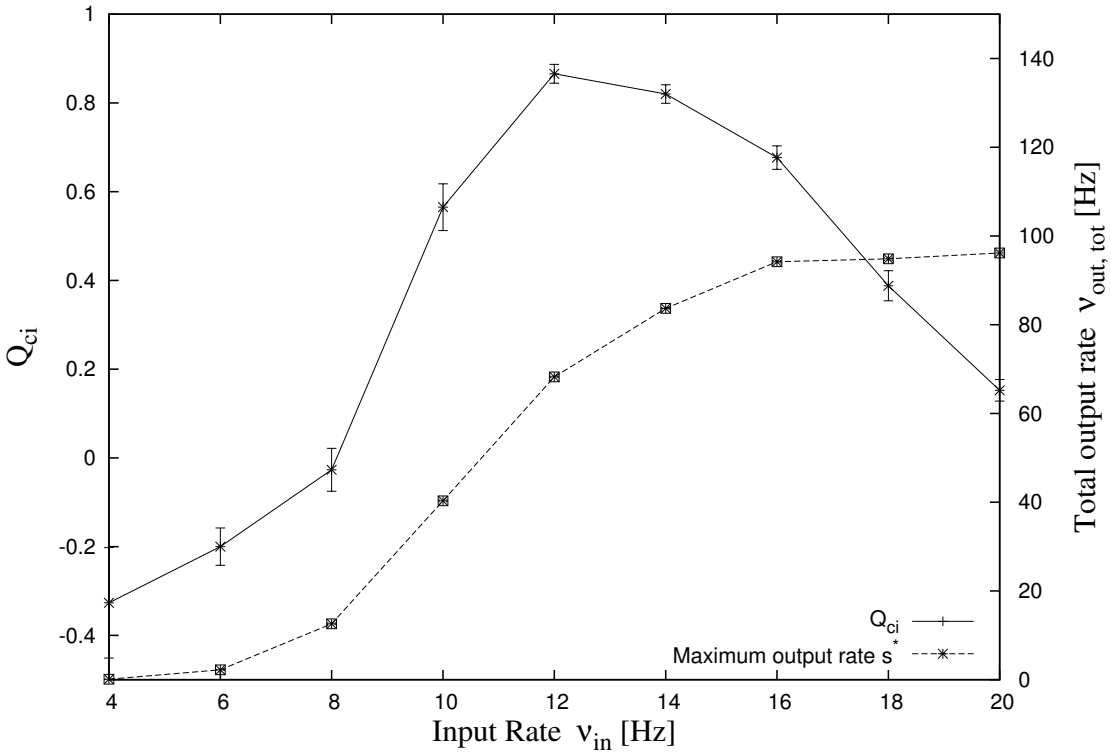
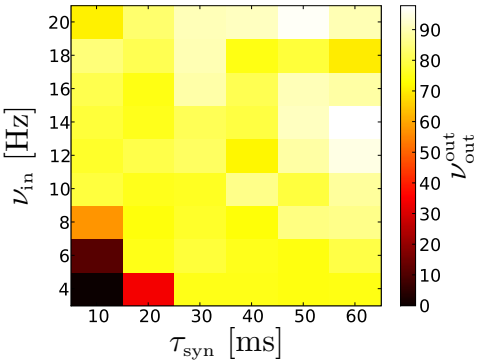
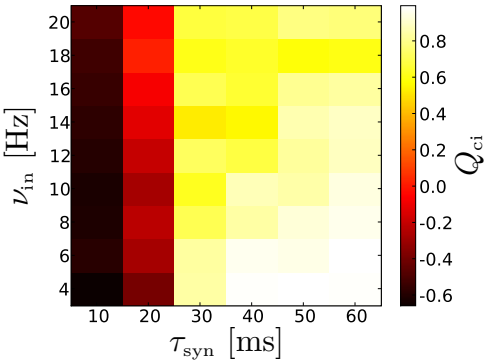


Figure 2.20: Solid line: Q_{ci}. Dotted line: $\nu_{\mathrm{out}}^{\mathrm{tot}}$. Shown are the mean values averaged over 50 runs with 5 s duration each. Error bars represent standard errors of mean and can not be seen for $\nu_{\mathrm{out}}^{\mathrm{tot}}$. Compare columns in figure 2.19b and 2.19a with τ_{syn}=30 ms. Strong excitation due to large ν_{in} counteracts mutual inhibition and hence decreases Q_{ci}.

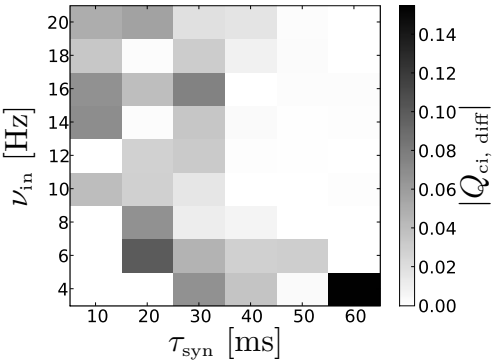


(a) Total output rate in a network of 16 neurons implementing cross inhibition.

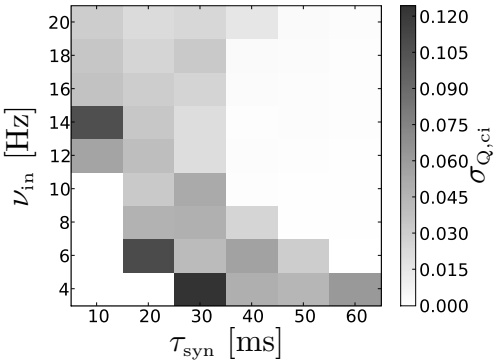


(b) Cross inhibition performance

Figure 2.21: Color map showing output rate and cross inhibition performance Q_{ci}, respectively, against synaptic time constant τ_{syn} and input rate ν_{in}. Each value is a mean value averaged over 50 runs with 5 s duration each. The number of excitatory and inhibitory inputs was adjusted between runs with different values for ν_{in} and τ_{syn}, so that $n_{\text{exc}} \cdot g_{\text{exc}} + n_{\text{inh}} \cdot g_{\text{inh}}$ is constant and the global output rate $\nu_{\text{out}}^{\text{tot}} \geq 75$ Hz. This is true except for small values of ν_{in} and τ_{syn}, where the external stimulation is not sufficient and the target global output rate of 75 Hz can not be reached with the given number of external stimuli. Figure 2.21b shows that the quality of cross inhibition performance is strongly dependent on the synaptic time constants.

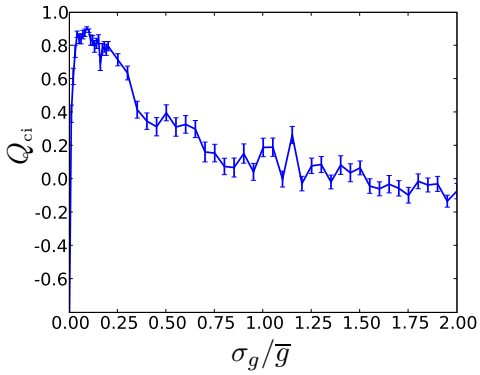


(a) The influence of quantized weights is shown by the absolute difference in Q_{ci} between a network with and one without quantized weights

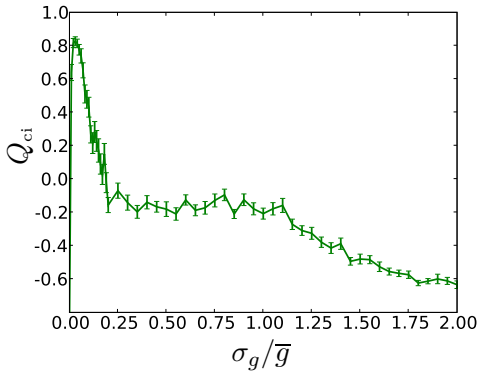


(b) Standard deviation of Q_{ci} in a network without quantized weights

Figure 2.22: Colormaps showing the standard error of mean of Q_{ci} in comparison to the effect of quantized weights on Q_{ci}. The color bar range was chosen to be the same in both plots.

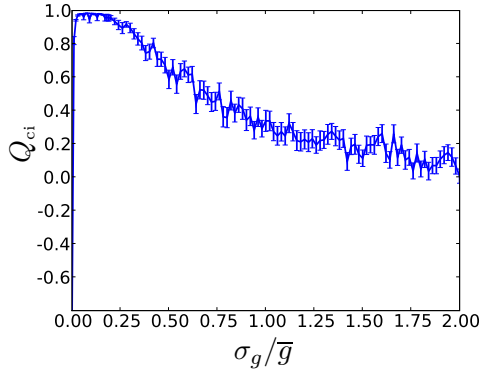


(a) Weight variation for external connections only

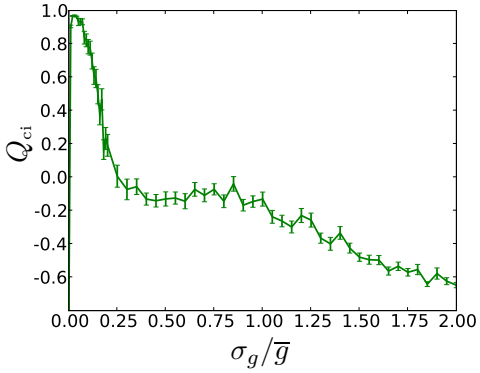


(b) Weight variation for feedback connections, too

Figure 2.23: Cross inhibition performance Q_{ci} in dependence on weight variation $\sigma_g/\overline{g}$ with $n_{\mathrm{exc}} = 48, g_{\mathrm{exc}} = 4{\cdot}10^{-4}\,\mu\mathrm{S}$. Each point represents a mean value of Q_{ci} averaged over 50 runs with 5 s duration each, with standard errors of mean as error bars. Blue curve (in fig. 2.23a) shows performance of a setup with uniform inhibitory feedback weights, i.e. all inhibitory connections have the same value as in table 2.1. Only the weights to external inputs are normally distributed. Green curve (in fig. 2.23b) shows Q_{ci} when additionally the feedback connections are normally distributed. The setup with normally distributed weights for feedback connections (fig. 2.23b) is more sensitive to $\sigma_g/\overline{g}$ which can be explained by fig. 2.25b.

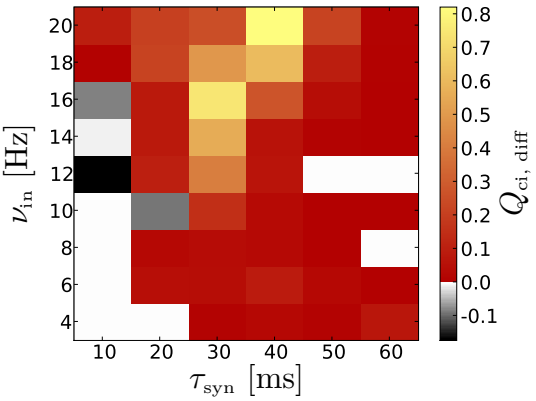


(a) Weight variation applied to external connections only.

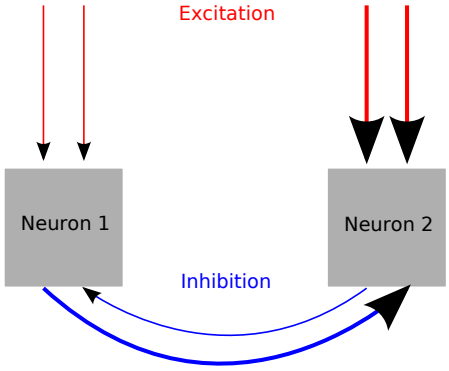


(b) Weight variation applied to external and to feedback connections.

Figure 2.24: Cross inhibition performance Q_{ci} in dependence on weight variation $\sigma_g/\overline{g}$ with twice the number of excitatory inputs but half the excitatory weight compared to the setup discussed above, see figure 2.23: $n_{\mathrm{exc}} = 96, g_{\mathrm{exc}} = 2 \cdot 10^{-4}\,\mu\mathrm{S}$.



(a) Difference of cross inhibition performance Q_{ci} between networks without and with normally distributed feedback weights. For the major part of the parameter range (plotted in gray scale), especially for high output rates (see figure 2.19b), Q_{ci} is larger when feedback weights are uniform. Positive values originate from statistical fluctuations. The global output rate is not adjusted, thus is as in figure 2.19a.



(b) Cross inhibition setup with 2 neurons and varying synaptic weights. Due to an unfavorable weight distribution the cross inhibition can not perform well in terms of a winner-take-all architecture. A neuron that is likely to fire because of strong excitation (neuron 2) can be strongly inhibited by another neuron (neuron 1). Thus, both are still able to fire and reduce Q_{ci}.

Figure 2.25: The variation of feedback weights can have a negative influence on Q_{ci}: Non-uniform weights for feedback connections decrease Q_{ci} significantly for ranges where cross inhibition does not work perfectly, i.e. $Q_{\text{ci}} < 1$ compare $\tau_{\text{syn}} = (30, 40)$ ms, $\nu_{\text{in}} \geq 12$ Hz in in figures 2.25a and 2.19b. An explanation can be found in an unfavorable weight distribution that penalizes neurons, that would dominate the output in a setup without varying feedback weights (see figure 2.25b). In ranges where either cross inhibition works well or the output rate is too small, feedback weight variation does not have a significant influence.

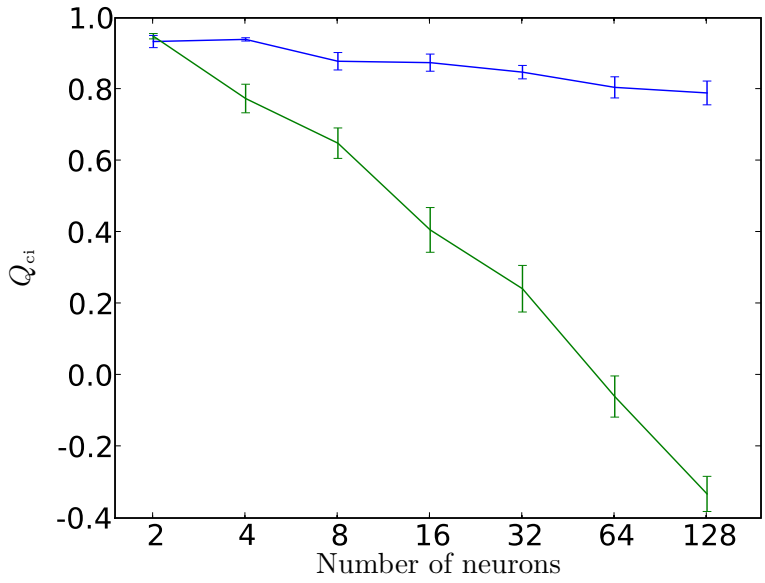
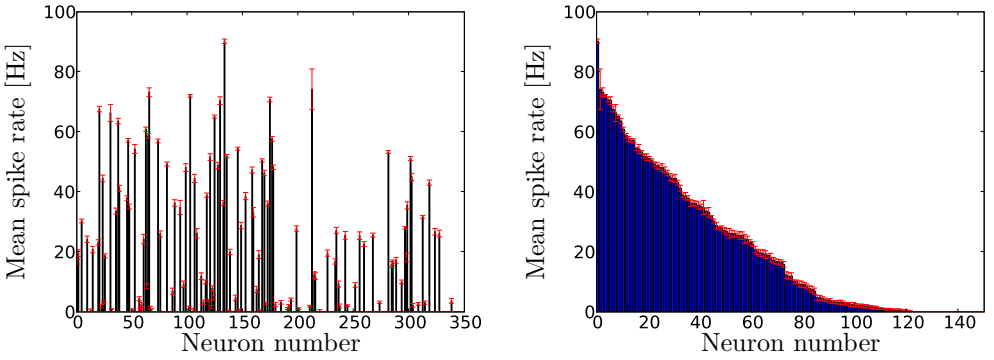


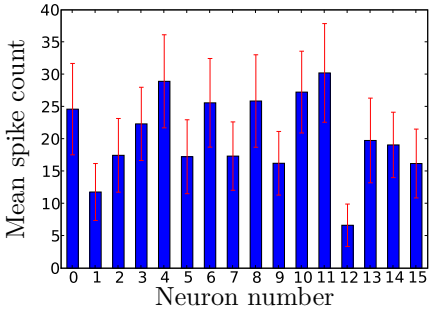
Figure 2.26: Cross inhibition performance is dependent on the network size. For each data point, 50 runs with 5 s duration each were simulated. Blue curve shows $Q_{ci}(N)$ in a setup with uniform feedback weights, whereas in the setup of the green curve feedback weights are normally distributed. Other parameters are according to table 2.1. Q_{ci} decreases with increasing network size due to two reasons: First, each neuron gets more inhibitory input and thus has more difficulties to prevail. Second, Q_{ci} decreases with growing total number of spikes fired by more than one neuron (see eq. 2.3).



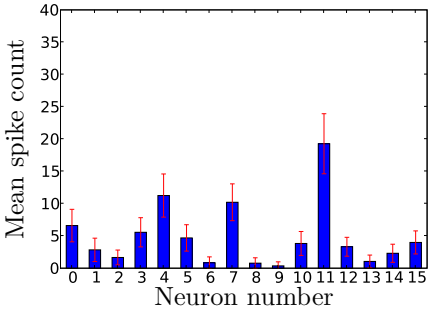
(a) Spike rate distribution over chip

(b) Distribution sorted by output rate

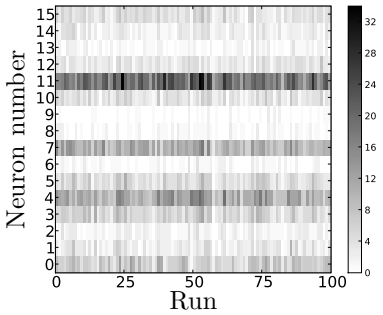
Figure 2.27: Mean spike rates averaged over 10 independent runs with 100 s duration each. Single neurons were stimulated with $n_{\text{exc}} = 48$ excitatory and $n_{\text{inh}} = 12$ inhibitory Poisson inputs with input rate $\nu_{\text{in}} = 10$ Hz. Both figures show the same data set. In order to allow a better comparison between neurons, the data shown in the right figure were sorted by decreasing output rate and neglecting silent neurons. Even after calibration of membrane time constants and synapse driver efficacies hardware neurons show a very inhomogeneous spiking behavior.



(a) Mean spike count of 16 hardware neurons



(b) Global cross inhibition



(c) Output spikecount with global cross inhibition over 100 runs

Figure 2.28: Spike count histograms on hardware without cross inhibition in comparison with two different cross inhibition setups. Shown is the mean spike count of 16 neurons on the hardware averaged over 100 runs. Error bars are standard deviations.

2.4 Self-organizing Winner Take All

Winner-take-all architectures are supposed to be one network architecture possibly implemented in cortical networks. There are many biologically realistic models of WTA architectures based on inhibitory neurons [*Elias and Grossberg*, 1975; *Amari and Arbib*, 1977; *Yuille and Grzywacz*, 1989; *Coultrip et al.*, 1992; *Kaski and Kohonen*, 1994]. Local WTA dynamics are supposed to be involved in visual velocity estimate [*Grzywacz and Yuille*, 1990] and regulate the total activity within networks with recurrent structure [*Johansson et al.*, 2002]. In the latter named, WTA modules are associated with hypercolumns, which is assumed to correspond to a cortical minicolumn and rather than an individual neuron. A modular structure of neural networks exhibit an improved noise tolerance, convergence speed and capacity [*Johansson et al.*, 2002]. One aspect that can contribute to draw a connection between the *syntax* of neural networks, e.g. firing patterns, to the *semantics* or meaning of them, is the idea that i.e. units of a hypercolumn, or in general one component taking part in WTA dynamics, represent one dedicated object or feature of the outside world. [*Johansson et al.*, 2002]. Furthermore, it is assumed that individual components or single neurons can be seen as invariant representations of objects from the real world [*Quiroga et al.*, 2005].

The computational power of the WTA architectures has been investigated mathematically by Maass in [*Maass et al.*, 2004]. Given there is also an overview of publications proposing effective VLSI implementations of WTA.

WTA circuits have been proposed amongst others for modeling selective attention [*Indiveri*, 2000] and making use of a learning rule for solving classification tasks [*Häfliger*, 2007].

Inspired by the latter named, this chapter presents a WTA architecture used for unsupervised, competitive learning experiments and investigates the portability of a similar WTA architecture to the hardware system.

2.4.1 Motivation

The winner-take-all architecture presented here is inspired by the work of [*Häfliger*, 2007]. It is attractive to be implemented on the FACETS Stage 1 hardware system because it is a basic application of STDP. The learning dynamics of which can be easily understood, so no misleading phenomena are expected to occur which might cause wrong implications, and it has been performed in different VLSI systems already.

Further practical reasons for the choice are that the problem size can be chosen such that it performs with only a small number of neurons, and that also the number of connections used for external stimulation and cross inhibition is sufficiently small to fulfill the limitations of the FACETS Stage 1 hardware system.

The application of the underlying learning rule (STDP) can show how sensitive or robust the performance of STDP is, with respect to process variations affecting the

synapse circuits. Thus, this experiment is intended to be a first simple benchmark for the implementation of STDP in the Stage 1 hardware system. The neuronal output is easily compared with the output that is expected from a successful learning process as computed by a reference simulation in NEST. Conclusions to be drawn from runs on the hardware system can reveal valuable insights which benefits the design of the STDP implementation in the Stage 2 hardware system.

2.4.2 Setup

The WTA setup presented here is based on strong mutual inhibitory connections and spike-dependent plasticity as a learning rule and is intended to perform pattern classification tasks after successful learning.

The WTA architecture utilized in the following experiments consists of a number of external inputs n_{in} each of which is connected via excitatory synapses to every N inhibitory neurons. These excitatory synapses obey the STDP rule which is explained in section 1.5. Initial weights for external stimulation must not be uniform, as this would stimulate all neurons in the same way and make all neurons spike at the same time. Thus, weights for external stimulation are normally distributed, just as in section 2.3.

The output of each neuron is connected to each other neuron via strong inhibitory synapses. In the following, this is called cross inhibition and is explained in detail in section 2.3. The inhibitory interconnections between all neurons do not follow the STDP rule, because an acausal firing of two neurons would depress the synapse between both and thus deteriorate the mutual inhibition, which would consequently impede the decision of a single winner. Consequently, inhibitory synapses are static in this setup. Figure 2.29 shows the schematic of a WTA architecture with $N = 4$ neurons, and $n_{\text{in}} = 4$ inputs.

In a perfectly working WTA architecture only one neuron is active at any given time. This can not be guaranteed for the whole progress of the experiment, because the excitatory stimulation varies with time and can lead to strong stimulation of multiple neurons.

Input:
The experimental setup at hand is structured as follows. Each input is connected to each neuron via the same number of synapses. The entire input population, consisting of n_{in} inputs, is subdivided into n_{P} disjunct sets of equal size. Each such set thus generates $n_{\text{spP}} = n_{\text{in}}/n_{\text{P}}$ independent Poisson spike trains. At any given moment in time, only one such set of neurons is active, while all others display no activity. This specific type of input, consisting of n_{spP} spike trains ($\nu \neq 0$) and $n_{\text{in}} - n_{\text{spP}}$ "silent" spike trains ($\nu = 0$) is, in the following, referred to as a pattern class, or more succinctly, a pattern (see figure 2.30) Without loss of generality, the active neurons that generate a pattern are considered adjacent. An example of such a pattern is shown in fig. 2.28. One element of a pattern class is presented to the neurons for a period of T_{P} with a subsequent pause

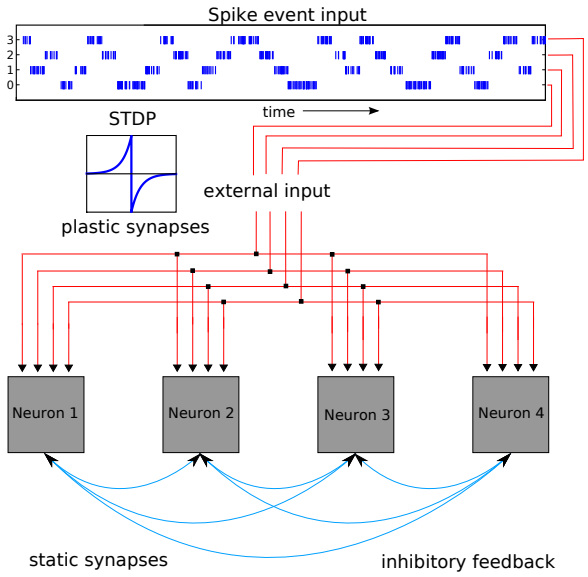


Figure 2.29: Schematic of a winner take all setup with $N = 4$ neurons and $n_{\text{in}} = 4$ inputs.

of 50 ms.

In a different pattern class n_{spP} other inputs hold the spike trains with $\nu_{\text{in}} > 0$. Thus, classes of patterns differ from each other by the disjoint set of active inputs and can be understood as n_{P}-dimensional binary vectors with one component unequal to zero. According to this, the number of pattern classes $n_{\text{P}} = n_{\text{in}}/n_{\text{spP}}$ has to break even, i.e. the set of inputs must be divided equally into n_{P} groups.

All spike trains with $\nu_{\text{in}} > 0$ are independent from each other. That holds for spike trains within one pattern class, as well as for different pattern classes and different cycles.

One cycle comprises n_{P} different pattern classes. Figure 2.31 shows the input for one cycle consisting of 4 pattern class samples. One cycle lasts $T_{\text{cycle}} = n_{\text{p}} \cdot T_{\text{p}}$. The whole experiment lasts n_{cycles} cycles. The order of pattern classes within one cycle is random.

Figure 2.32 shows the input spike trains comprising five cycles.

Intended result after learning

The network is intended to classify n_{P} different classes of patterns through the activity of the N neurons. In other words: After successful learning, the activity of one certain neuron represents the existence of one certain class of pattern in the input.

This can be realized by a learning process, which is described in the following: The subset of inputs representing one class of patterns stimulates all neurons at the same time as long as the input is present at the input. If this stimulation makes one neuron spike,

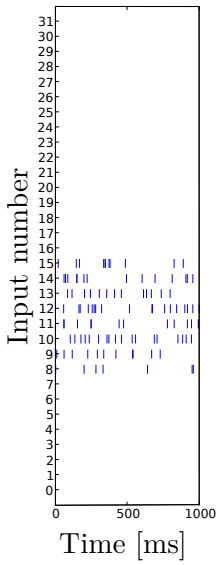


Figure 2.30: Spike trains representing one element of a pattern class. $n_{\mathrm{spP}} = 8$ different Poisson spike trains stimulate the neuron population for $T_{\mathrm{P}} = 1000$ ms with a certain mean rate ν_{in}. The classes of input patterns can be understood as n_{P}-dimensional binary vectors with one component unequal to zero. Here, e.g. $\vec{v} = (0, 1, 0, 0)$. On that level of abstraction, one can use the term pattern, as the binary vector symbolizing the presented spike trains is repeated in time, even though the exact structure of the spike trains is not repeated.

the synaptic weights of connections between the neuron and the subset are strengthened according to the STDP rule given in section 1.5. Additionally, the neuron sends an action potential to all other neurons via its inhibitory connections and in that way tries to impede the other neurons from spiking. The neuron that gains the strongest weight modification during that pattern class will have highest probability to fire the next time when the same pattern class is fed into the network. In the best case scenario, this happens for a different neuron in each different pattern.

After a successful learning procedure, the output signal of one neuron represents the one class of input signals that is currently fed into the network. One example of a successful learning procedure is shown in figure 2.33.

If one neuron is activated by more than one subset of input, it can easily happen that this neuron learns to represent more than one class of pattern. Furthermore, it can happen that the connections between one subset and more than one neuron are strengthened. This would lead to a response of more than one neuron, if that class of pattern is presented. This is not the desired behavior of the network.

2.4.3 Performance Measures for WTA

In the WTA architecture described above, synapses transmitting external stimuli obey the STDP learning rule described in section 1.5. Thus, the amount of external stimulation one neuron receives is not constant in time, especially at the beginning of the experiment. Consequently, one can not presume that cross inhibition works perfectly all the time, i.e. that only one neuron is active at the same time. This can be seen in figure

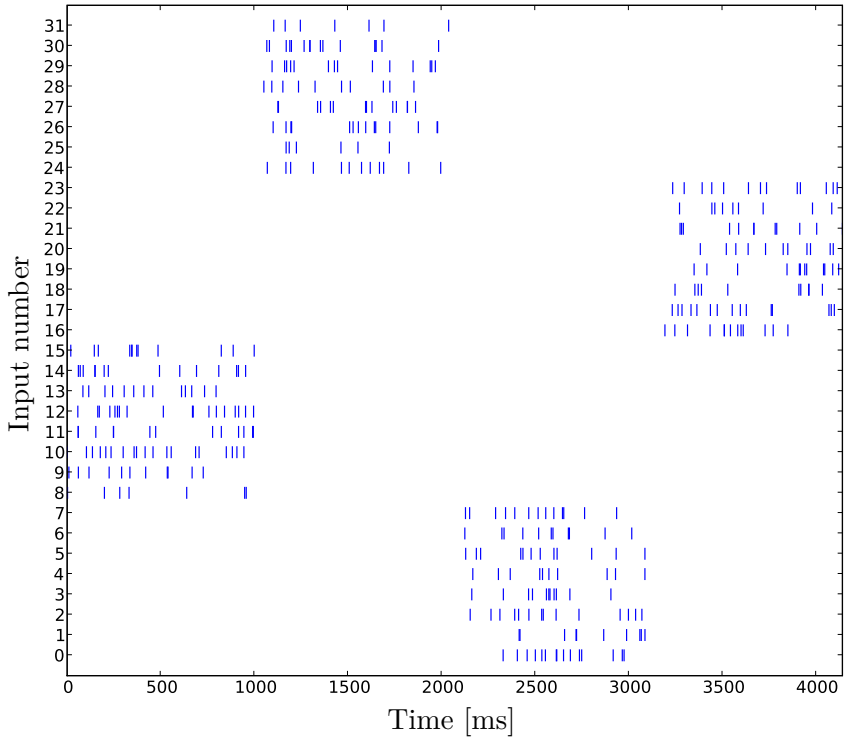


Figure 2.31: Spike trains representing the input for one cycle. $n_{\mathrm{spP}} = 8$ different Poisson spike trains represent one pattern class and stimulate the neurons at the same time. In this case, one cycle comprises elements from 4 different pattern classes. One cycle has a duration of 4 pattern durations plus an interval between each pattern.

2.34a, where all neurons spike during the first learning cycle.

In order to evaluate the efficacy of mapping the disjoint binary input vectors to the output spike trains, one has to assess the dissimilarity between output spike trains from different neurons over all pattern presentations within one cycle *and* the clearness of a single pattern. One preferable representation to monitor the learning process is a mapping from the output spike trains to a vector. That vector can then be used to measure the quality of learning.

One approach taken by Häfliger in [*Häfliger*, 2007] is to assume N possible output states (one for each neuron) and apply the classical definition of entropy on the output

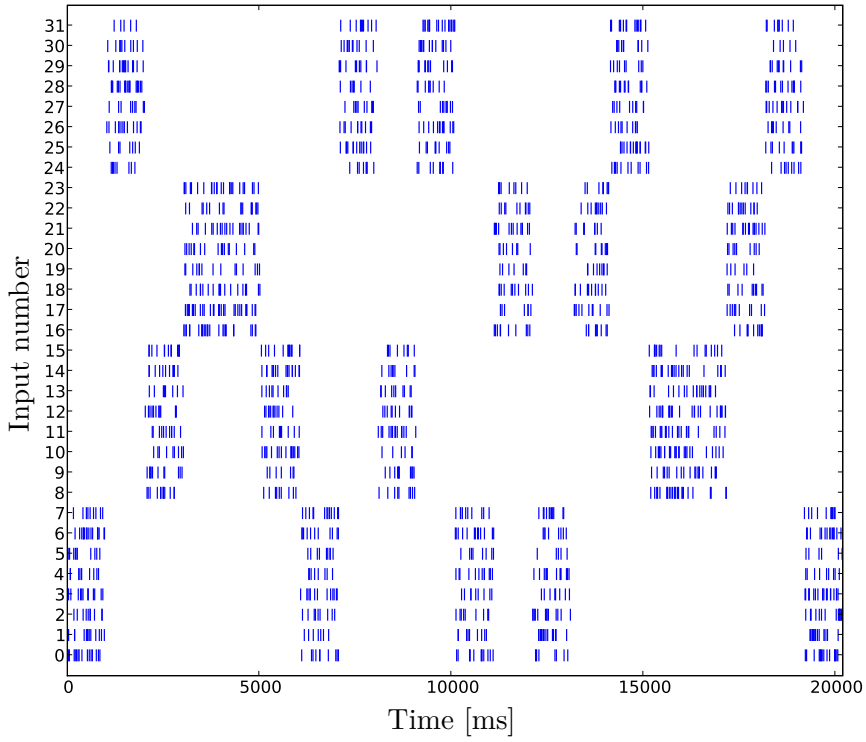


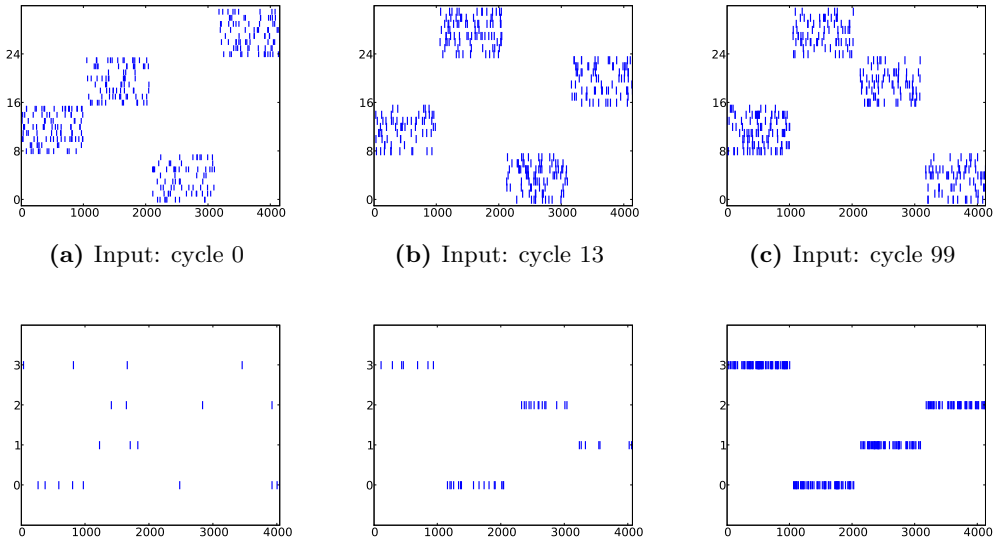
Figure 2.32: Spike trains showing the input for five cycles. In each cycle one element of the 4 different pattern classes is fed into the network.

states to measure the quality of encoding:

$$H_{\mathrm{WTA}} = -\sum_{i=0}^{N-1} p_i \cdot \log_2 p_i \tag{2.6}$$

where p_i is the probability of the i^{th} neuron being the winner. The probabilities p_i were computed as the sum of the probabilities of the input patterns to which they responded. (cf. [*Häfliger*, 2007]) In the case that two neurons would respond when presented with only one input pattern, the contribution to their activity probability is split between both according to their respective number of output spikes during that period.

Given the case that not only two, but all neurons respond during one input pattern presentation, which occurs in the simulations performed here, the entropy calculated above is not an adequate measure for quantifying the ability of the setup to distinguish different spike patterns. This can be clarified by the following example result with four



(a) Input: cycle 0 **(b)** Input: cycle 13 **(c)** Input: cycle 99

(d) Output: cycle 0, $Q_{\text{wta}} = 0.667, \nu_{\text{out}}^{\text{tot}} = 4.76\,\text{Hz}$ **(e)** Output: cycle 13, $Q_{\text{wta}} = 4.0, \nu_{\text{out}}^{\text{tot}} = 10.7\,\text{Hz}$ **(f)** Output: cycle 99, $Q_{\text{wta}} = 3.84, \nu_{\text{out}}^{\text{tot}} = 55\,\text{Hz}$

Figure 2.33: Rasterplots showing input and output during a successful learning procedure. At the beginning, the output rate is low and increases in course of the experiment. Notice that the first spike in cycle 0 is fired neuron 3 (highest row in fig. 2.33d). After 100 cycles neuron 3 represents input vector $\vec{v} = (0, 1, 0, 0)$, neuron 0 represents $\vec{v} = (0, 0, 0, 1)$, and so on. This outcome is already visible in cycle 13 (middle column).

output spike trains. Supposing that for each pattern all neurons produce the same number of output spikes (see figure 2.35c), the probability p_i of a neuron to be the winner is calculated as follows:

$$p_i = \frac{1}{n_{\text{P}}} \cdot \sum_{j=0}^{n_{\text{P}}} \frac{s_j(i)}{S_j}$$

where n_{P} is the number of different pattern, $s_j(i)$ is the number of output spikes of neuron i during pattern j and S_j is the total number of output spikes during pattern j.

During the period of any pattern the respective number of output spikes is one fourth of the total spike output, thus $s_j(i)/S_j = 1/4$. This leads to a maximum entropy $H = 2$ even for a worst case scenario where all neurons fire the same number of spikes during each pattern. This is why it makes no sense to regard the output spike trains as discrete states and measure the quality of encoding as done above, when there is a possibility that several neurons can be active during the same pattern.

Consequently, entropy is not an adequate measure because it only contains the number of possible outcomes and does not discern the shape of these different states. To use entropy with the above defined probabilities p_i is not practical for the above mentioned reasons.
Here is a detailed computation for the example outputs shown in figure 2.35 according to eq. 2.6: 2.35a: $p_0 = p_1 = p_2 = p_3 = 1/4$
$H = -(\frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2(1/4)) = 2$

2.35b: $p_3 = p_2 = 1/4$, $p_1 = 1/4 + 1/2 \cdot 1/4 = 3/8$, $p_0 = 1/8$
$H = -(\frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2(1/4) = 1.906$

2.35a: $p_0 = p_1 = p_2 = p_3 = \sum_{i=0}^{n_\mathrm{P}=4} 1/4 \cdot 1/4 = 1/4$
$H = -(\frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2(1/4) + \frac{1}{4}\log_2(1/4)) = 2$

The problem with the approach using the classical entropy definition is that output spike trains do not represent discrete states like neuron is active or not, because one additionally has to take the output spike trains from other neurons into account. Thus, the number of possible output states can not be estimated in a simple way.

A different approach that accounts for the case that several neurons are active during the same pattern presentation period is based on the quality measure for cross inhibition introduced in section 2. The approach presented here tries to answer the question how many different patterns occur in the output which are represented by one single neuron. In order to answer this question one first has to quantify the response given to the set of patterns with respect to the winner-take-all aspect.

In the following a measure that quantifies the number of distinct output patterns created by the activity of single neurons is presented. This measure concentrates on the winner-take-all condition, i.e. one neuron is assigned to not more than one pattern and its contribution to the measure is higher the more it dominates the others.

The presented measure counts the number of responses to a pattern class with respect to the clearness of the networks' response. The response to a given pattern is regarded as clear, when only one neuron responds while all other neurons are silent during that pattern presentation.

A network that performs the classification problem successfully has to give preferably unambiguous responses to the presented input patterns. How distinct the response to one class of pattern is, can be estimated by a slightly modified version of equation 2.3:

$$\begin{aligned} Q^+_{\mathrm{ci}} &= \frac{2s^* - S}{S} \quad \text{if} \quad \frac{2s^* - S}{S} > 0 \\ Q^+_{\mathrm{ci}} &= 0 \qquad \text{else} \end{aligned} \tag{2.7}$$

Q^+_{ci} is positive when the output spike count of the most active neuron s^* is larger than

half of the total output spike count, i.e. $s^* > S/2$. Provided the case two neurons produce an output too similar, such that $\frac{2s^*-S}{S}$ would become less than zero, Q_{ci}^+ is set zero, which is corresponds to an ambiguous response of the classifier.

To get an answer to the question how many patterns have been distinguished by the network, the clearness of a single pattern response *and* the dissimilarity between different pattern responses have to be taken into account. This is done by the following procedure:

For each pattern presentation period j , Q_{ci}^+ is calculated. The value is set to element (i, j) in a response matrix as shown in figure 2.4.3, whereas i is the index of the most active neuron during pattern presentation period j. The quality of the classifier's response in that cycle is estimated by summing over maxima of Q_{ci}^+ for each neuron:

$$Q_{\text{wta}} = \sum_{i}^{N} \max_{j=[0..n_{\text{P}}]} \{Q_{\text{ci}}^+\} \tag{2.8}$$

The defined measure satisfies all requirements of the following analysis. It is maximized by a successful classification with a clear-cut winner, and is decreased by both an increase of the ambiguity of various neurons' responses and by cases where one neuron responds most to multiple patterns. Hence, although the measure can not be interpreted as the number of possible distinct outputs per input pattern set, it is well suited to monitor the development of classification ability of a winner-take-all network via STDP. The factor 2 in equation 2.7 is appropriate for the small network architectures analyzed in the following, but should be replaced by a carefully chosen, larger value when switching to larger networks.

Another important aspect that has to be considered is the development of weights during the learning process. In total there are $N \cdot n_{\text{in}}$ plastic weights. As the large number of individual weights does not obviously reveal information about the global learning process, mean weights are regarded. Weights can be averaged over all $N \cdot n_{\text{in}}$ connections which yields $\langle w_{ij} \rangle$ or weights can be averaged in two steps: First for each neuron i the length of the weight vector $\mid \vec{w}_i \mid$ is calculated, then this length is averaged over all neurons. For simplicity reasons the first option will be used,

Concluding, the following analysis will use two quantities to monitor the learning process of the network: Q_{wta} and the mean weight change $\langle w_{ij} \rangle$.

2.4.4 Simulation Parameters and Results

The setup described above is determined by the following parameters:

- N: number of neurons in the network
- n_{spP}: number of synapses per pattern
- n_{P}: number of pattern classes presented to the network
- n_{cycles}: number of learn cycles
- n_{in}: number of excitatory inputs, $n_{\text{in}} = n_{\text{spP}} \cdot n_{\text{P}}$

- ν_{in}: frequency for pattern generation
- g_{exc}: weights for excitatory synapses (external stimulation)
- g_{feedback}: weights for inhibitory feedback synapses
- $\sigma_g/\overline{g}$: relative weight variation parameter for external synapses, i.e. width of the Gaussian distribution relative to the mean value of the distribution ($\overline{g} \neq 0$)

Additionally there is a number of parameters determining the learning process:

- A_+: amplitude of the positive branch of the STDP curve
- A_-: amplitude of the negative branch of the STDP curve
- τ_+: time constant for the positive branch of the STDP curve
- τ_-: time constant for the negative branch of the STDP curve
- $g_{\max}$: maximum value for weight update function

Similar to the cross inhibition setup, it is a priori not clear which set of parameter values results in the best performance. In this section, only a subset of parameters is studied. To study the impact of a certain parameter, the experiments were simulated 50 times with different random seeds. The learning process was monitored by averaging Q_{wta} and the mean weight change $\langle w_{ij} \rangle$ over 50 runs. One of the first questions that has to be answered concerns the stabilization of the network, i.e. when weights cease to change significantly. This is an important question, because the answer determines the required length of a learning experiment and the number of learn cycles n_{cycles}. The temporal change of synaptic weights is monitored over 100 cycles using the following generic parameter set:

Basic parameters		
N	4	
n_{spP}	8	
n_{P}	4	
n_{cycles}	100	
n_{in}	32	
ν_{in}	11	Hz
g_{exc}	$12 \cdot 10^{-4}$	μS
g_{feedback}	$120 \cdot 10^{-4}$	μS
$\sigma_g/\overline{g}$	0.1	
A_+	0.0	
A_-	0.012	
$\tau_+ = \tau_-$	10	ms
$g_{\max}$	$4 \cdot g_{\text{exc}}$	

Figure 2.37a shows the development of the mean weight change of one synapse, i.e. one weight vector component, between two successive cycles. The mean synapse weight

was averaged over all 128 synapses, and then averaged over 50 experiments. The plot shows that the mean synapse weight change first increases till cycle number 30 and then decreases over a period of about 50 cycles to zero. Due to limited number of trials, several outliers have an impact on the mean development.
There are several important issues to be regarded:

1) The maximum weight change of one synapse is in the order of magnitude of 1%.
2) After about 80 cycles the mean weight change equals zero.
3) Mean weight changes are in general positive.

To 1): The small change of a synapse weight between cycles makes a possible application of this setup on the hardware system more realistic, because the minimum weight change in hardware is one bit, i.e. 6.25 %, which would would be accumulated over ca. 6 cycles in simulation. 6 cycles take 6 s + $6 \cdot 4 \cdot 50$ ms for the pauses between each pattern presentation. The weight update in the hardware system is done periodically with a period depending on the number of plastic synapse rows (see section 1.5). If only a subset of synapse rows is used (ca. 1/6), the weight update period matches the expected period for the accumulation of causal correlations. Thus, only few correlations occur that have no effect, because the weight update flag is already set.
2): An experiment duration of 100 cycles is sufficient.
3): Causal correlations outweigh acausal ones. Consequently, the output rate increases, which can already be seen in the exemplary plots shown above (see figure 2.33). Issue number 3) will be discussed in subsection 2.4.5 with respect to normalization.

Instead of the change of individual synaptic weights between two cycles, figure 2.37a also shows the actual mean synaptic weight.

Figure 2.38 shows the learning progress. Shown is the performance measure Q_{wta} defined above versus number of cycles. Again, the plotted data is the mean value out of 50 experiments with the same basic parameter set given above.

The final value for Q_{wta} which represents the number of patterns that can be successfully classified by the neuron population is ca. 2.72 ± 0.2. This result can be explained as follows: It often occurs, that one neuron learns to respond to two classes of input patterns and hence decreases Q_{wta}. This happens, e.g. because of a certain weight distribution promotes one neuron to be the winner when two patterns are present at the input. This increase of weights of one neuron happens at the expense of another neuron, which has smaller initial weights and hence is not able to fire correlated spikes. Another reason might be an ineffective cross inhibition. The strength for mutual inhibition was chosen to be nearly two times stronger compared with the basic parameters in section 2.3 to minimize the probability of coincident responses. Further studies are necessary to study the strength of mutual inhibition.

One possibility to compensate the imbalance of initial weight is to increase the number of synapses per pattern n_{spP}. A repartition of the external stimulus showed a small positive effect on the performance of cross inhibition as shown in section 2.3.3. Figure

2.39b shows that this effect does not lead for a better performance of the learning experiment. The plot shows Q_{wta} against the number of learn cycles. The increased number of inputs does not result to a significant increase in Q_{wta}, but a slower convergence to the final value.

Other parameters that influence the convergence speed of the learning process are $\nu_{\mathrm{in}}, A_-, A_+$ and w_{max} Figure 2.39a shows the impact of larger amplitudes of the STDP learning rule. The plot shows that Q_{wta} increases very rapidly at the beginning, but does not converge to a final value within the simulated time. Presumably, this is caused by strong fluctuations induced by the larger amplitudes A_-, A_+, which are ten times higher in this simulation compared to the basic parameter set given above. Consequently, the total number of learn cycles has to be chosen according to A_-, A_+.

2.4.5 Self-organized WTA: Discussion and Conclusion

The presented simulation results yield that the number of cycles after which the synaptic weights reach their final value is about 80 for the chosen parameter values, depending on the input parameters. As the first studies do no provide satisfying results, several parameters which would increase the complexity of the setup have not been investigated, e.g. the number of neurons in the network and the number of patterns presented to the network.

The number of synapses per pattern was increased by a factor of 4 in order to counteract imbalances in the initial distribution of weights, which would lead to a privileging of single neurons. An increased number of synapses does not lead to a better performance, but a longer duration needed for stabilization.

The amplitudes of the weight modification in the STDP learning rule A_-, A_+ also well have an impact on the speed with which Q_{wta} converges to the final value. It was found, that too high values of A_-, A_+ can lead to instability and a worse performance. Further parameters that affect the required number of learn cycles are ν_{in} and g_{max}. Their effect was not analyzed in detail, because a variation of both could not increase Q_{wta} significantly.

With regard to the implementation of the presented setup onto the hardware, the most important aspect is the non-continuous weight update. The accumulated weight modification function is read out periodically, with a maximum update period of 45.6 s in biological time if all synapse rows are used as plastic. The update period is a crucial parameter, as it decides whether correlated spikes are discarded or not. This can happen, when the update period is too large and the accumulated weight modification function has passed the necessary threshold. This can lead to different results than expected:

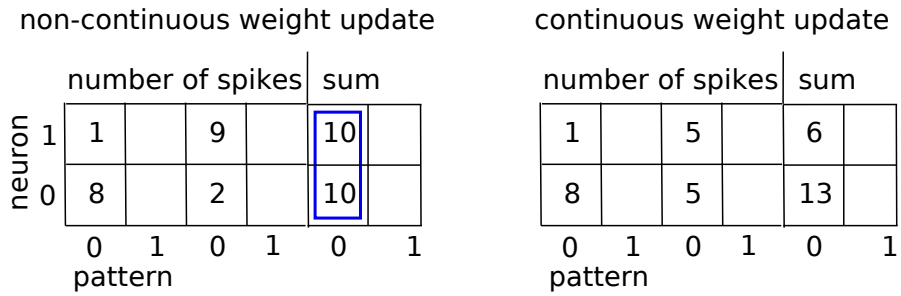


Figure 2.40: A non-continuous weight update leads to different results compared with a continuous weight update. Assumed the weight update flag is set, if ≥ 10 spikes detected. Left: Two weight updates. Right: One

The exact value of the update period determines other parameters as e.g. the amplitudes of the modification function A_-, A_+, the desired spike rate and hence the parameters defining the external stimulation $(\nu_{\text{in}}, g_{\text{exc}}, n_{\text{spP}})$. Thus, this parameter requires proper adjustment.

After these considerations, porting the proposed setup to the hardware will presumably cause problems and lead to different results than the software simulations show.

Normalization One important disadvantage of the applied learning rule is that weight vectors of each neuron are not normalized. This means, that the length of weight vectors converging to the same neuron change during the learning procedure. The absence of normalization prevents competition between synaptic weights w_{ij} that belong to the same post-synaptic neuron i. Competition in this context means a decrease of weights belonging to all other synapses, if one synapse is increased. If weight normalization is implemented, the synaptic weights w_{ik} between one neuron i and inputs representing one pattern class would decrease in the case, that this neuron does not represent the corresponding pattern class. This would ease the situation for other neurons to learn this pattern class and hence would make it more difficult for one neuron to learn more than one pattern class. A discussion of weight normalization can be found e.g. in [*Gerstner and Kistler*, 2002].

The results presented above do not make use of weight normalization, which is presumably one reason for the frequent case that one neuron represents more than one pattern class after the learning procedure. A normalization of weight vectors could be implemented in the hardware system, by reading out the weight memory and modifying the weights according the desired normalization rule. The quantization of weight in this context allows only very coarse modifications, which would have a much stronger impact than normalization done with continuous weights.

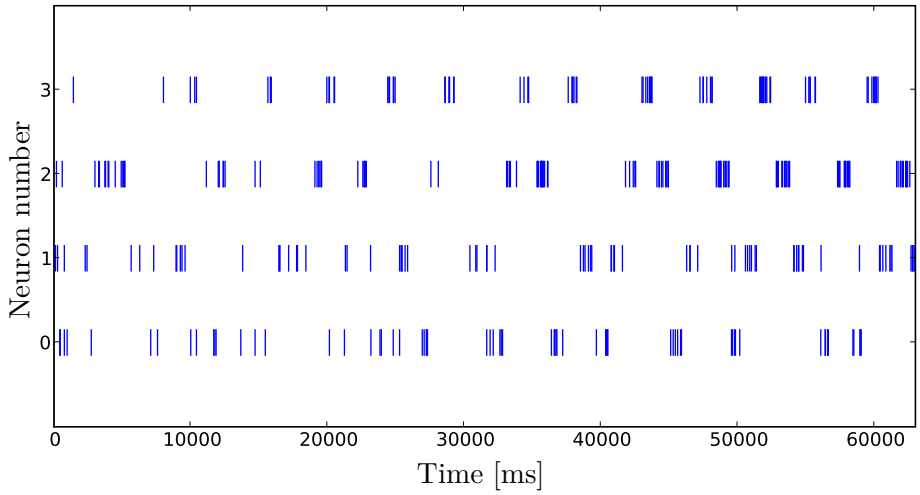
One possible application of the presented architecture could be a principal component analysis (PCA) of the input space. For this purpose, input spike trains with different

frequencies are shown to the network with the goal that after learning, responses only occur for the classes of spike trains with the highest frequency.

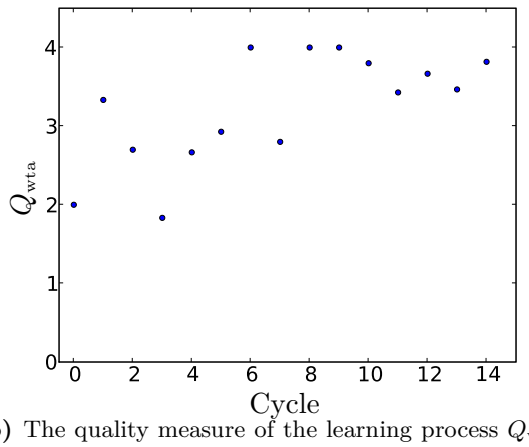
2.5 Hardware Implementation

A hardware implementation of the presented self-organizing winner-take-all architecture is still not possible. Multiple obstacles avoid a porting of the setup to the FACETS Stage 1 hardware system version 3: On a representable set of synapses using a single parameter set, STDP curves have been measured [*Müller*, 2008], but a consistent behavior was not yet establishable. In the current development status, a mapping of model parameters (amplitudes A_+, A_-, time constants τ_-, τ_+) to hardware parameters is still missing, since extensive measurements for a wide parameter range are work in progress. After these studies will be finished, the STDP software layer stack has to be completed, thus enabling the standard PyNN STDP interface. This includes flexible access to synaptic weights and translation of model parameters to hardware parameters.

Furthermore, as was shown in section 2.3, the essential mechanism of cross-inhibition performs poorly on the current version of the hardware, mainly due to parasitic threshold-reset inter-dependencies. Hence, in order to port a winner-take-all architecture as implemented in section 2.4.4 to the FACETS Stage 1 hardware system, at least these parasitic phenomena have to be eliminated or massively minimized. Additionally, the configurability of the STDP feature has to be established, which requires further investigations in cooperation with the designers of the system.

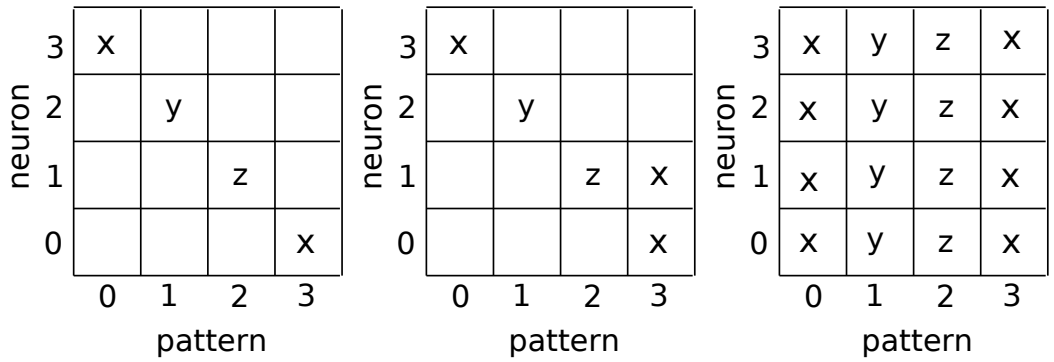


(a) Rasterplot showing spike output during a successful classification



(b) The quality measure of the learning process Q_{wta} is defined in equation 2.8 and corresponds to the output shown above.

Figure 2.34: One example of a successful pattern classification. The upper figure shows the response to 4 different patterns injected through $4 \cdot 8$ synapses. Shown are the first 15 cycles. Each pattern is 1000 ms long with a subsequent pause of 50 ms, thus one cycle lasts 4200 ms. The lower figure shows the quality measure defined in equation 2.8.



(a) Output entropy $H = 2$ **(b)** $H = 1.906$ **(c)** $H = 2$

Figure 2.35: Schematic of 3 possible output spike counts. The x-axis represents the dimension of time, the y-axis the neuron number. The matrix elements represent the number of spikes fired by the neuron during the corresponding pattern presentation. In figure 2.35a only one neuron is active at any given time, thus the probability of each neuron is 1/4 and the output entropy reaches its maximum value $H = 2$. In figure 2.35b, the total number of output spikes during pattern 3 is evenly split between neuron 1 and neuron 0, which leads to a decreased output entropy of $H = 1.906$. Figure 2.35c shows a worst case scenario where cross inhibition does not work at all and all neurons fire the same number of spikes during each pattern. Even though the all neurons carry the same amount of information during each pattern, the entropy as calculated in eq. 2.6 is the same as in the best case $H = 2$.

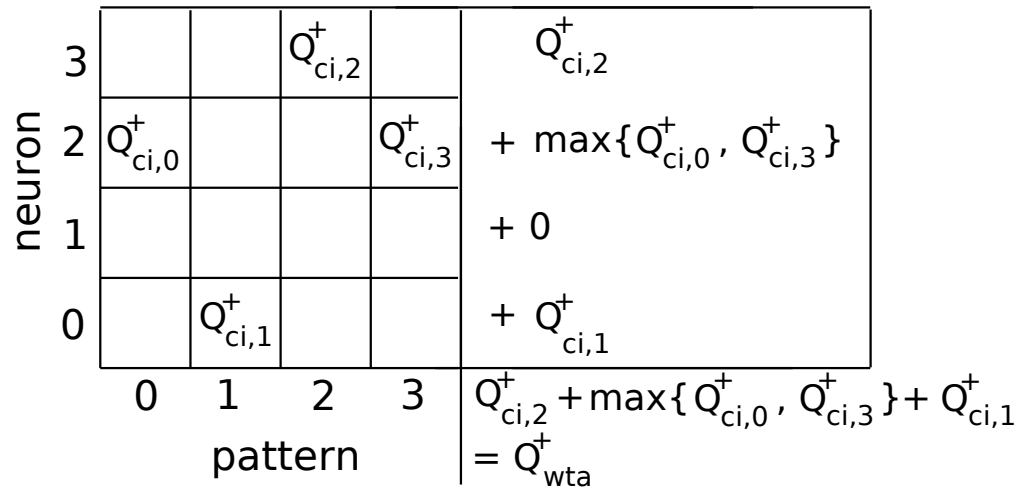
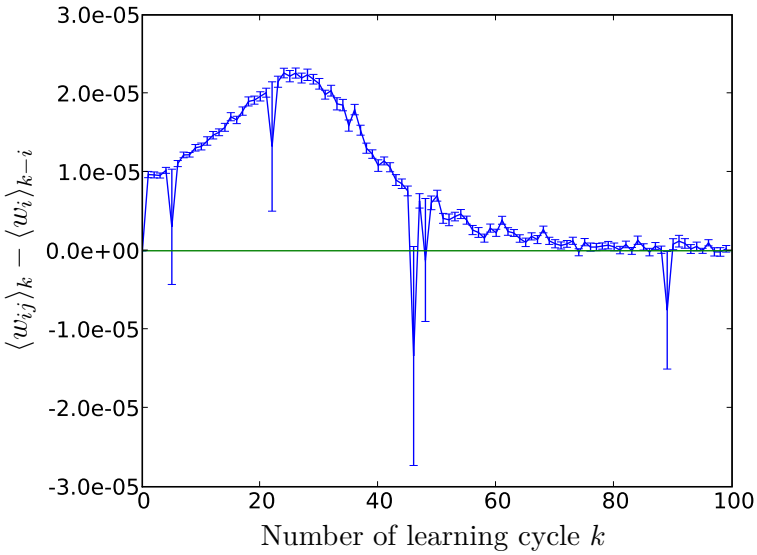
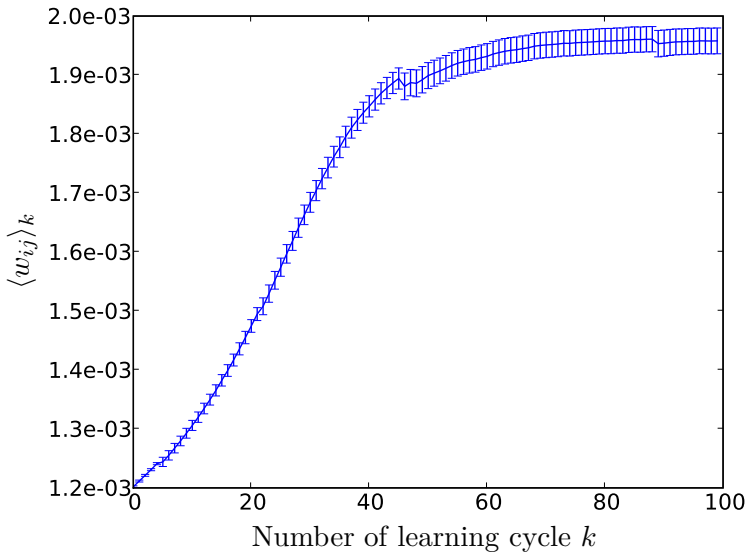


Figure 2.36: Example for a response matrix with N rows and n_P columns. Q^+_ci is calculated for each pattern and set to (i, j). i is the row of neuron that is most active during pattern j. During presentation of pattern 0, neuron 2 was most active. At the end of the cycle, Q_wta is calculated as the sum of the maximum $Q^+_{\mathrm{ci,\,i}}$ from all neurons $i = [0..N]$.



(a) WTA: Mean difference of weight vector component $\langle w_i \rangle$ between two cycles in the learning process.



(b) Development of the mean synapse weight averaged over all synapses

Figure 2.37: Development of synaptic weights in the learning process. Synaptic weights are averaged over all synapses and over 50 simulations with different seed. Error bars represent the standard error of mean originating from averaging over 50 experiments. The mean difference of a synaptic weight between two cycles is in the order of magnitude of 1% of the actual value. After about 80 cycles weights have converged to their final value. Thus, the total length of the learning experiments is set to 100 cycles.

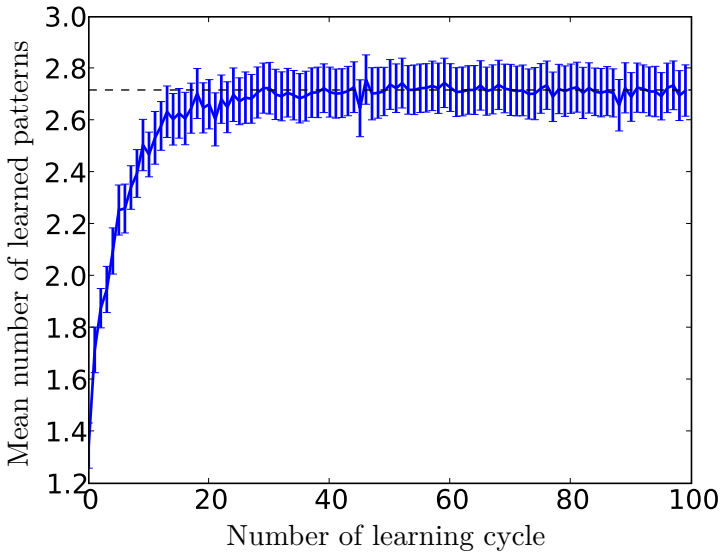
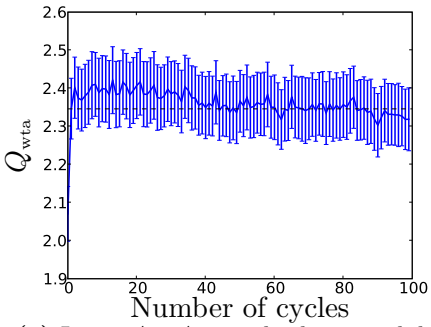
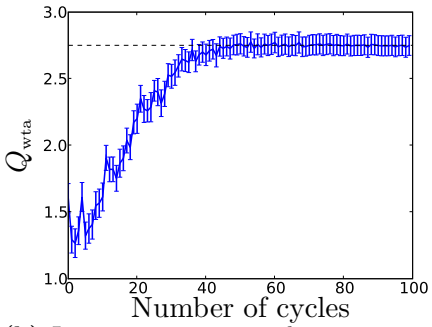


Figure 2.38: $Q_{\text{wta}} \sim$ Mean weight vector component development during the learning process. Mean number of learned patterns averaged over 50 runs in dependence of the learning cycle. Error bars represent the standard error of the mean originating from averaging over 50 runs. Interestingly, the performance reaches the final value already after 30 cycles, which is the time of the maximum weight change according to figure 2.37a, even though the mean synaptic weight does not reach its final value until 80 cycles (see figure 2.37b).



(a) Large A_+, A_- can lead to instability



(b) Learn experiment with more synapses per pattern

Figure 2.39: Mean number of learned patterns averaged over 50 runs in dependence of the learning cycle. Error bars represent the standard error of the mean out of 50 experiments. The left figure shows an experiment with $A_- = 0.10, A_+ = 0.12$, which is ten times larger than the basic parameter set (compare figure 2.38). At first, Q_{wta} increases rapidly and reaches a maximum a few cycles. Then, Q_{wta} decreases and does not saturate within the simulated time range. This is due to large fluctuations imposed by the large amplitudes of the learning rule. The right figure shows the average performance over 50 runs with an increased number of synapses per pattern $n_{\mathrm{spP}} = 32$, instead of 8 compared to the basic parameter set. To keep the amount of excitatory stimulation constant and investigate only the influence of the increased number of synapses, the synaptic weights were divided by a factor of 4 to $g_{\mathrm{exc}} = 3$. The final value of Q_{wta} is not higher compared to fig. 2.38, i.e. the network does not learn better when the input is distributed over more, but weaker inputs. However, the time required for stabilization of weights is higher.

Discussion and Outlook

The use of neuromorphic hardware offers new perspectives for modeling and investigating long-term self-organization principles of the brain. Especially in the FACETS project, the implementation of STDP in conjunction with the highly accelerated operation can face the challenge proposed by increasing complexity of neural network models.

In order to apply these features, the following approach was taken: A simple self-organizing experiment was presented which offers the possibility to adjust the complexity level of the task to be learned. Then, occurring problems were analyzed step by step and solutions for the realization of the experiment were presented. One step towards the realization is the effective mutual inhibition. The influence of various parameters on a cross inhibition setup was investigated, and, among other insights, it was found that synaptic time constants have to be chosen with care. For quantitative analysis of the cross inhibition performance, a measure was introduced and successfully applied. In order to measure hardware synaptic time constants, indirect methods have to be used, since the only possible observables in the hardware system are spike output and membrane potential. Together with spike-triggered averaging, it is straight-forward to estimate synaptic time constants if a neuron is in a high-conductance state. In order to determine a neuron's conductance state, a method was introduced and proved to be well working in software simulations and in the hardware. The proposed method allows to estimate the temporal resolution capability of a neuron in a purely spike-based way. Thus, the transition to a high-conductance state can be identified and therefore a regime can be determined in which the measurement of synaptic time constants works.

Applying this test, synaptic time constants were measured in the hardware system and turned out to be coarsely adjustable within the range of 20 ms to 60 ms. Transferring the cross inhibition setup turned out to be very difficult: Non-uniform spiking thresholds among neurons were tried to be compensated by the adjustment of individual weights. But this technique turned out to be insufficient. A specific calibration of inhibitory synapse drivers is necessary because an insufficient effective mutual inhibition presumably is one important reason for the bad performance on the hardware. Another important improvement can be achieved by increasing the number of input synapses. This is beneficial because more weights can be used for adjustment in order to counteract imbalances. A population based approach, including short-term plasticity, with the aim to compensate variations in the substrate, requires further studies until it could be implemented.

A winner-take-all architecture utilizing weight manipulation mechanisms, similar to

those in the hardware, was investigated in software simulations. Even though self-organized learning increased the performance in terms of the proposed quality measure, no parameter set was found which reliably led to successful classifications. Only in a small fraction of setups, the desired maximum output quality was achieved. An important aspect to be considered for software as well as for hardware applications, is a normalization of weights, which introduces the required competition between synapses. An effective implementation of weight normalization in hardware synapses is difficult to realize due to the quantized weights. Furthermore, possible realizations would require additional inner-chip communication, because synapses would need to exchange mutual information about their current weight. A space-efficient solution in each hardware synapse is hard to realize and not expected in the near future.

The execution of the winner-take-all experiment in hardware was not possible, because a reliable STDP mechanism was not available during this thesis. Since cross inhibition is essential for the realization of the proposed winner-take-all architecture, it is highly recommended to successfully implement cross inhibition on the hardware, first.

Generally, the obstacles arising when trying to use the hardware for a seemingly simple self-organization experiment are non-trivial. Setups, that easily work in software due to perfectly controllable dynamics and parameters, can fail badly on a system with intrinsic noise, transistor-level variations and other inhomogeneities or parasitic parameter interdependencies.

Similar phenomena are found in biological systems, i.e. the brain, which obviously manage to cope with such effects and perform well, anyhow. Efforts to implement experiments (like the WTA setup) provide a fruitful way in searching for strategies that can avoid, counterbalance or average out such inhomogeneities (e.g. population based solutions or self-stabilizing architectures [*Bill*, 2008]). Each solution found on this path can provide new tools, methods and insights that help to understand and improve the system in general. The presented high-conductance state test is one example for such a solution.

Since several chip inherent bugs and inaccuracies have been identified, many of which will be fixed in a future revision of Stage 1, it is likely that subsequent steps can successfully be realized on this hardware substrate. New principles, such as population coding, could balance the variations in single components. The upcoming Stage 2 of the FACETS hardware system will provide the necessary foundation to investigate such promising and resource-intensive approaches.

A Appendix

A.1 Simulation Parameters

For all experiments in this thesis which utilized the NEST software simulator, the NEST neuron model called *iaf_cond_exp_sfa_rr* was applied. It implements a conductance-based leaky integrate-and-fire model with exponentially decaying conductance courses in its synapses, with the option of spike frequency adaptation and relative refractoriness. The last two features were disabled throughout all simulations presented, so the software models resembles the idealized hardware model as described in equation 1.1. If not stated differently in the setup description of an experiment, the following parameters were used:

Parameter	Formulaic Notation	PyNN Notation	Value	
Membrane Capacitance	C_{m}	cm	200.0	nF
Reset Potential	V_{reset}	v_reset	-80.0	mV
Inhibitory Reversal Potential	E_{I}	e_rev_I	-75.0	mV
Resting Potential	E_l	v_rest	-70.0	mV
Threshold Voltage	V_{thresh}	v_reset	-57.0	mV
Excitatory Reversal Potential	E_{E}	e_rev_E	-0.0	mV
Membrane Leakage Conductance	g_{l}	g_leak	20.0	nS
Refractory Period	τ_{ref}	t_ref	1.0	ms
Excitatory Synapse Time Constant	$\tau_{\mathrm{syn,E}}$	tau_syn_E	30.0	ms
Inhibitory Synapse Time Constant	$\tau_{\mathrm{syn,I}}$	tau_syn_I	30.0	ms

Table A.1: Neuron and synapse model parameters as used throughout this thesis (if not explicitly stated differently).

Bibliography

Abrahams, D., and R. Grosse-Kunstleve, Building hybrid systems with boost.python, 2003.

Amari, S., and M. Arbib, Competition and cooperation in neural nets, in *Systems Neuroscience*, edited by J. Metzler, New York: Academic Press, 1977.

Amemori, K., and S. Ishii, Effect of the synaptic time constant on stochastic spiking neurons, in *7th International Conference on Neural Information Processing (ICONIP)*, vol. 1, pp. 6–11, 2000.

Barlow, R. J., *Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences (The Manchester Physics Series)*, reprint ed., John Wiley & Sons Ltd, 1989.

Bi, G., and M. Poo, Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type, *Neural Computation*, *9*, 503–514, 1997.

Bienenstock, E. L., L. N. Cooper, and P. W. Munro, Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex, *Journal of Neuroscience*, *2*, 32–48, 1982.

Bienenstock, E. L., L. N. Cooper, and P. W. Munro, *Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex*, MIT Press, Cambridge, MA, USA, 1988.

Bill, J., Self-stabilizing network architectures on a neuromorphic hardware system, Diploma thesis (English), University of Heidelberg, 2008.

Bontorin, G., S. Renaud, A. Garenne, L. Alvado, G. Le Masson, and J. Tomas, A real-time closed-loop setup for hybrid neural networks, in *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS2007)*, 2007.

Boustani, S. E., M. Pospischil, M. Rudolph-Lilith, and A. Destexhe, Activated cortical states: experiments, analyses and models, *Journal of Physiology (Paris)*, *101*, 99–109, 2007.

Braden, R. T., RFC 1122: Requirements for Internet hosts — communication layers, 1989.

Brette, R., and W. Gerstner, Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity, *J. Neurophysiol.*, *94*, 3637 – 3642, 2005, article.

Brüderle, D., personal communication, 2008.

Brüderle, D., A. Grübl, K. Meier, E. Mueller, and J. Schemmel, A software framework for tuning the dynamics of neuromorphic silicon towards biology, in *Proceedings of the 2007 International Work-Conference on Artificial Neural Networks (IWANN'07)*, vol. LNCS 4507, pp. 479–486, Springer Verlag, 2007.

Buzsaki, Feed-forward inhibition in the hippocampal formation, *Progress in Neurobiology*, *22*, 131–153, 1984.

Buzsaki, C., Temporal structure in spatially organized neuronal ensembles: a role for interneuronal networks, *Current Opinion in Neurobiology*, *5*, 504–510, 1995.

Buzsaki, G., M. Penttonen, Z. Nadasdy, and A. Bragin, Pattern and inhibition-dependent invasion of pyramidal cell dendrites by fast spikes in the hippocampus in vivo, *Proceedings of the National Academy of Sciences of the United States of America*, *93*, 9921–9925, 1996.

Caporale, N., and Y. Dan, Spike timing-dependent plasticity: A hebbian learning rule., *Annual review of neuroscience*, 2008.

Chow, C., Phase-locking in weakly heterogeneous neuronal networks, *Physica D*, *118*, 343–370(28), 1998.

Cossart, R., D. Aronov, and R. Yuste, Attractor dynamics of network up states in the neocortex, *Nature*, *423*, 238–283, 2003.

Coultrip, R., R. Granger, and G. Lynch, A cortical model of winner-take-all competition via lateral inhibition, *Neural Netw.*, *5*, 47–54, 1992.

Dan, Y., and M. Poo, Spike timing-dependent plasticity of neural circuits, *Neuron*, *44*, 23–30, 2004.

Davison, A., PyNN – a python package for simulator-independent specification of neuronal network models, `http://www.neuralensemble.org/PyNN`, 2008.

Davison, A., D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, PyNN: a common interface for neuronal network simulators, *Frontiers in Neuroinformatics*, *pending published*, 2008.

Dayan, P., and L. F. Abott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, The MIT press, Cambride, Massachusetts, London, England, 2001.

de Boer, R., and P. Kuyper, Triggered correlation., *IEEE Trans Biomed Eng*, *15*, 169–179, 1968.

Destexhe, A., Conductance-based integrate-and-fire models, *Neural Comput.*, *9*, 503–514, 1997.

Destexhe, A., M. Rudolph, and D. Pare, The high-conductance state of neocortical neurons in vivo, *Nature Reviews Neuroscience*, *4*, 739–751, 2003.

Diesmann, M., and M.-O. Gewaltig, NEST: An environment for neural systems simulations, in *Forschung und wisschenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis 2001*, edited by T. Plesser and V. Macho, vol. 58 of *GWDG-Bericht*, pp. 43–70, Ges. für Wiss. Datenverarbeitung, Göttingen, 2002.

Drake, F. L. (Ed.), *Python Reference Manual: February 19, 1999, Release 1.5.2*, iUniverse, Incorporated, 2000.

Ehrlich, M., C. Mayr, H. Eisenreich, S. Henker, A. Srowig, A. Grübl, J. Schemmel, and R. Schüffny, Wafer-scale VLSI implementations of pulse coupled neural networks, in *Proceedings of the International Conference on Sensors, Circuits and Instrumentation Systems (SSD-07)*, 2007.

Elias, S. A., and S. Grossberg, Pattern formation, contrast control, and oscillations in the short term memory of shunting on-center off-surround networks, *Biol. Cybern.*, *20*, 69–98, 1975.

FACETS, Fast Analog Computing with Emergent Transient States, project homepage, `http://www.facets-project.org`, 2008.

Fieres, J., A. Grübl, S. Philipp, K. Meier, J. Schemmel, and F. Schürmann, A platform for parallel operation of VLSI neural networks, in *Proc. of the 2004 Brain Inspired Cognitive Systems Conference (BICS2004)*, University of Stirling, Scotland, UK, 2004.

Fieres, J., J. Schemmel, and K. Meier, Realizing biological spiking network models in a configurable wafer-scale hardware system, in *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008.

Gerstner, W., and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, 2002.

Gewaltig, M.-O., and M. Diesmann, NEural Simulation Tool NEST (Scholarpedia article), `http://www.scholarpedia.org/article/Nest`, 2008.

Grübl, A., VLSI implementation of a spiking neural network, Ph.D. thesis, Ruprecht-Karls-University, Heidelberg, 2007, document No. HD-KIP 07-10.

Grzywacz, N. M., and A. L. Yuille, *A model for the estimate of local velocity*, vol. 427/1990, Springer Berlin / Heidelberg, 1990.

Häfliger, P., Adaptive WTA with an analog VLSI neuromorphic learning chip, *IEEE Transactions on Neural Networks*, *18*, 551–72, 2007.

Hansel, D., G. Mato, and C. Meunier, Synchrony in excitatory neural networks, *Neural Comput.*, *7*, 307–337, 1995.

Hebb, D. O., *The Organization of Behaviour*, Wiley, New York, 1949.

Holmes, W. R., and W. B. Levy, Insights into associative long-term potentiation from computational models of NMDA receptor-mediated calcium influx and intracellular calcium concentration changes, *Journal of Neurophysiology*, *63*, 1148–1168, 1990.

IEEE, Standard for information technology - portable operating system interface (POSIX). shell and utilities, *Tech. rep.*, IEEE, 2004.

Indiveri, G., Modeling selective attention using a neuromorphic analog vlsi device, *Neural Comput.*, *12*, 2857–2880, 2000.

Izhikevich, E. M., Polychronization: Computation with Spikes, *Neural Comp.*, *18*, 245–282, 2005.

Johansson, C., A. S, and A. Lansner, A neural network with hypercolumns, in *In LNCS: Vol. 2415. Proceedings of the international conference on artificial neural networks*, 2002.

Kaski, S., and T. Kohonen, Winner-take-all networks for physiological models of competitive learning, *Neural Networks*, *7*, 973–984, 1994.

Koch, C., *Biophysics of Computation: Information Processing in Single Neurons*, Oxford University Press, 1999.

Kumar, A., S. Schrader, A. Aertsen, and S. Rotter, The high-conductance state of cortical networks, *Neural Computation*, *20*, 1–43, 2008.

Legenstein, R., C. Naeger, and W. Maass, What can a neuron learn with spike-timing-dependent plasticity?, *Neural Computation*, *17*, 2337–2382, 2005.

Levy, W., and O. Steward, Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus, *Neuroscience*, *8*, 791–97, 1983.

Maass, W., T. Natschläger, and H. Markram, On the computational power of circuits of spiking neurons, *Journal of Physiology (Paris)*, *(in press)*, 2004.

Markram, H., Y. Wang, and M. Tsodyks, Differential signaling via the same axon of neocortical pyramidal neurons., *Proceedings of the National Academy of Sciences of the United States of America*, *95*, 5323–5328, 1998.

Matsumura, M., D.-f. Chen, T. Sawaguchi, K. Kubota, and E. E. Fetz, Synaptic Interactions between Primate Precentral Cortex Neurons Revealed by Spike-Triggered Averaging of Intracellular Membrane Potentials In Vivo, *J. Neurosci.*, *16*, 7757–7767, 1996.

Mead, C. A., *Analog VLSI and Neural Systems*, Addison Wesley, Reading, MA, 1989.

Mead, C. A., and M. A. Mahowald, A silicon model of early visual processing, *Neural Networks*, *1*, 91–97, 1988.

Merolla, P. A., and K. Boahen, Dynamic computation in a recurrent network of heterogeneous silicon neurons, in *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, 2006.

Morrison, Abigail, Diesmann, Markus, Gerstner, and Wulfram, Phenomenological models of synaptic plasticity based on spike timing, *Biological Cybernetics*, *98*, 459–478, 2008.

Morrison, A., C. Mehring, T. Geisel, A. Aertsen, and M. Diesmann, Advancing the boundaries of high connectivity network simulation with distributed computing, *NeuralComput*, *17*, 1776–1801, 2005.

Morrison, A., A. Aertsen, and M. Diesmann, Spike-Timing-Dependent Plasticity in Balanced Random Networks, *Neural Comp.*, *19*, 1437–1467, 2007.

Müller, E., Operation of an imperfect neuromorphic hardware device, Diploma thesis (English), University of Heidelberg, 2008.

Muller, E. B., Markov process models for neural ensembles with spike-frequency adaptation, Ph.D. thesis, Ruprecht-Karls University Heidelberg, 2006.

Neves, G., S. F. Cooke, and T. V. Bliss, Synaptic plasticity, memory and the hippocampus: a neural network approach to causality, *Nat Rev Neurosci*, *9*, 65–75, 2008.

Oja, E., A simplified neuron model as a principal component analyzer, *Journal of Mathematical Biology*, *15*, 267–273, 1982.

Pfister, J.-P., T. Toyoizumi, D. Barber, and W. Gerstner, Optimal Spike-Timing-Dependent Plasticity for Precise Action Potential Firing in Supervised Learning, *Neural Comp.*, *18*, 1318–1348, 2006.

Philipp, S., A. Grübl, K. Meier, and J. Schemmel, Interconnecting VLSI Spiking Neural Networks Using Isochronous Connections, in *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN'2007)*, vol. LNCS 4507, pp. 471–478, Springer Verlag, San Sebastián, Spain, 2007.

Prescott, S. A., S. Ratte, Y. De Koninck, and T. J. Sejnowski, Nonlinear interaction between shunting and adaptation controls a switch between integration and coincidence detection in pyramidal neurons., *J Neurosci*, *26*, 9084–97, 2006.

Quiroga, Q. Q., L. Reddy, G. Kreiman, C. Koch, and I. Fried, Invariant visual representation by single neurons in the human brain, *Nature*, *435*, 1102–1107, 2005.

Renaud, S., J. Tomas, Y. Bornat, A. Daouzli, and S. Saïghi, Neuromimetic ics with analog cores: an alternative for simulating spiking neural networks, in *Proceedings of the 2007 IEEE Symposium on Circuits and Systems (ISCAS2007)*, 2007.

Rudolph, M., and A. Destexhe, Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies, *Neural Comput.*, *18*, 2146–2210, 2006.

Schemmel, J., A. Grübl, K. Meier, and E. Mueller, Implementing synaptic plasticity in a VLSI spiking neural network model, in *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN'06)*, IEEE Press, 2006.

Schemmel, J., D. Brüderle, K. Meier, and B. Ostendorf, Modeling synaptic plasticity within networks of highly accelerated I&F neurons, in *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS'07)*, IEEE Press, 2007.

Schemmel, J., J. Fieres, and K. Meier, Wafer-scale integration of analog neural networks, in *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008.

Serrano-Gotarredona, R., M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, H. K. Riis, T. Delbrück, and S.-C. Liu, AER building blocks for multi-layer multi-chip neuromorphic vision systems, in *Advances in Neural Information Processing Systems 18*, edited by Y. Weiss, B. Schölkopf, and J. Platt, pp. 1217–1224, MIT Press, Cambridge, MA, 2006.

Shelley, M., D. McLaughlin, R. Shapley, and J. Wielaard, States of high conductance in a large-scale model of the visual cortex, *J. Comp. Neurosci.*, *13*, 93–109, 2002.

Song, S., K. Miller, and L. Abbott, Competitive hebbian learning through spiketiming-dependent synaptic plasticity, *Nat. Neurosci.*, *3*, 919–926, 2000.

Sussillo, D., T. Toyoizumi, and W. Maass, Self-Tuning of Neural Circuits Through Short-Term Synaptic Plasticity, *J Neurophysiol*, *97*, 4079–4095, 2007.

Terman, D., and D. Wang, Global competition and local cooperation in a network of neural oscillators, *Phys. D*, *81*, 148–176, 1995.

The Neural Simulation Technology (NEST) Initiative, Website, `http://www.nest-initiative.org`, 2008.

Tsodyks, M., and H. Markram, The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability, *Proceedings of the national academy of science USA*, *94*, 719–723, 1997.

Vogelstein, R. J., U. Mallik, J. T. Vogelstein, and G. Cauwenberghs, Dynamically reconfigurable silicon array of spiking neuron with conductance-based synapses, *IEEE Transactions on Neural Networks*, *18*, 253–265, 2007.

Vreeswijk, C. V., Partial synchronization in populations of pulse-coupled oscillators, *Physical Review E*, *54*, 5522–5537, 1996.

Watts J, T. A., Excitatory and inhibitory connections show selectivity in the neocortex, *Journal of Physiology*, *562*, 89–97, 2005.

Worgotter, F., and B. Porr, Temporal Sequence Learning, Prediction, and Control: A Review of Different Models and Their Relation to Biological Mechanisms, *Neural Comp.*, *17*, 245–319, 2005.

Yuille, A. L., and N. M. Grzywacz, A winner-take-all mechanism based on presynaptic inhibition feedback, *Neural Comput.*, *1*, 334–347, 1989.

Acknowledgments (Danksagungen)

Ich möchte mich bei allen bedanken, die zum Gelingen dieser Arbeit beigetragen haben. Dies sind insbesondere:

Herrn Prof. Dr. Karlheinz Meier und Dr. Johannes Schemmel für die freundliche Aufnahme in die Arbeitsgruppe und die stets hilfreiche Unterstützung.

Alain Destexhe für die Übernahme des Zweitgutachtens.

Thanks to all the *Softies* for the massive support!!! and for all the ... well ... you know ...

Ganz besonderer Dank gilt meiner Familie für die allumfassende Unterstützung.

Statement of Originality (Erklärung):

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, December 10, 2008

.....
(signature)