Faculty of Physics and Astronomy

University of Heidelberg

Diploma thesis

in Physics

submitted by

Johannes Bill

born in Frankfurt a.M.

Dezember 2008

Self-Stabilizing Network Architectures on a Neuromorphic Hardware System

This diploma thesis has been carried out by Johannes Bill at the
Kirchhoff-Institute for Physics
University of Heidelberg
under the supervision of
Prof. Dr. Karlheinz Meier

Abstract

Self-Stabilizing Network Architectures on a Neuromorphic Hardware System

This thesis presents methods to improve the usability of a neuromorphic hardware device. The utilized chip physically implements a network of spiking neuron models. It is operated with a high acceleration compared to biological real-time and is designed for the investigation of computational principles inspired by the brain. Its application is hindered by characteristics of the implemented units, as emulation results reflect inhomogeneities within the utilized substrate. In a first step, various sources of imperfection are identified, specified and, if possible, counterbalanced by calibration routines. In order to further increase the homogeneity of the substrate, balancing approaches on the network level are sought. Extensive software simulation studies prepare the adoption and successful application of biologically inspired self-stabilizing architectures to the hardware system. It turns out that the application of short term synaptic plasticity is vital for achieving a foundation the research on brain-like computing with neuromorphic hardware can build upon.

Selbst-Stabilisierende Netzwerk-Architekturen auf einem neuromorphen Hardwaresystem

Die vorliegende Arbeit zeigt Methoden zu einer Verbesserung der Benutzbarkeit eines neuromorphen Hardwaresystems auf. Der verwendete Mikrochip implementiert ein Netzwerk spikender Neuronenmodelle. Im Vergleich zur biologischen Echtzeit läuft das System mit stark erhöhter Geschwindigkeit und ist für die Untersuchung am Gehirn orientierter Prinzipien der Informationsverarbeitung bestimmt. Die Anwendung wird durch individuelle Merkmale der verwendeten Komponenten erschwert, da sich Inhomogenitäten des zugrundeliegenden Substrats in den Emulationsergebnissen widerspiegeln. Zunächst werden verschiedene Störfaktoren ermittelt, spezifiziert und, falls möglich, mit Hilfe von Kalibrierungsprozessen abgeschwächt. Zur weiteren Verbesserung der Substrathomogenität werden ausgleichende Konzepte auf Netzwerkebene gesucht. Die Übertragung und erfolgreiche Anwendung biologisch motivierter selbststabilisierender Architekturen auf die Hardware wird durch umfangreiche Softwaresimulationen vorbereitet. Die Verwendung synaptischer Kurzzeitplastizität stellt sich dabei als wesentlich für das Erreichen einer Grundlage heraus, auf der die Erforschung am Gehirn orientierter Informationsverarbeitung aufbauen kann.

Contents

Introduction

Today, personal computers and custom information processing devices – like mobile phones, digital cameras or pocket calculators – are well-established in everyday life. Interestingly, they follow a completely different design principle than another, natural and powerful system: the mammalian and, in particular, the human brain. Although these man-made utilities reveal an astounding capabilities, many tasks are obviously performed much more efficiently by a human mind. Simply, a phenomenological regard roughly divides the different skills of both models:
The biological approach outclasses any technical system concerning recognition of faces or speech as well as almost every learning assignment. The easier a problem can be solved by means of a straight algorithm, the more powerful SISD-architectures[1] gain. In 1996, for the first time in history the chess computer 'Deep Blue' won a game against the world champion Garry Kasparov. Finally, when it comes to basic arithmetics, computers out-paced men a long time ago.

So, the question arises, how these differences can be characterized on a basic level. In the late 19th century, it was discovered that the brain consists of nerve cells – so-called *neurons* – which are interconnected by *synapses*.[2] Today, we know that the human brain inherently processes information in parallel, while yielding a high fault tolerance regarding malfunctions of individual components. The *cerebral cortex*, essentially involved in many intelligent functions of the brain, was found to exhibit a widely homogeneous structure, independent of the competence of a specific area [*Mountcastle*, 1979]. The universality of the cortical tissue is underlined by the ability of its domains to adopt functions of other domains, e.g. after an accident, and by the similarity of its texture in different species [*Lehmann and Löwel*, 2008]. Furthermore, the cortex features self-tuning of its activity [*Shu et al.*, 2003; *Destexhe et al.*, 2003; *Boustani et al.*, 2007] and organizes its synaptic connections without supervision [*Bi and Poo*, 1997; *Dan and Poo*, 2004].

The question on what properties of the involved units are of particular importance and on how the different units need to be interconnected in order to yield their unique computational power is easily formulated. The answer to this issue touches various scientific disciplines, like biology, medicine, mathematics, computer science or psychology. This diversity is reflected in the participants of the *FACETS project*,[3] which is funded by the European Commission within the *Future Emergent Technology* program. The FACETS

[1] SISD: Single Instruction, Single Data stream. For instance, the 'von Neumann computer'.
[2] The human brain contains approximately 10^{11} neurons and 10^{14} synapses.
[3] FACETS: Fast Analog Computing with Emergent Transient States

members comprise 13 groups of 7 European countries including biologists, computer scientists, engineers and physicists, aiming at *"conceiving paradigms of computation that depart significantly from the Turing concept of contemporary IT systems and make use of the complex and ongoing dynamics seen in brain activity."* [*FACETS*, 2008].
Understanding the human brain will lead to promising medical therapies as well as new computing devices and is likely to form one of the major achievements of the 21th century.

As a natural science, the investigation of this computational architecture is based on measurements and modeling. Thus, new techniques, which allow the investigation of the basic components – the neurons and synapses – on a microscopic level, boosted knowledge in neuroscience, considerably. For instance, the patch clamping technique [*Sakmann and Neher*, 1995] provided access to single ion channel dynamics of neuron membranes. Consequently, profound models of neuron dynamics [*Hodgkin and Huxley*, 1952; *Brette and Gerstner*, 2005] and synaptic plasticity [*Tsodyks and Markram*, 1997; *Markram et al.*, 1998] were developed.
A different approach investigates higher brain regions via *electroencephalography* (EEG) or *functional magnetic resonance imaging* (fMRI) [*Huettel et al.*, 2004]. The obtained insights allow to identify the role of different domains within the brain and provide valuable indicators on how information is processed (see e.g. [*Hubel and Wiesel*, 1962, 1965]).
Nevertheless, a vital point of the brain capacity lies in its connection structure. Each neuron is synaptically linked to thousands of other neurons, forming a complex recurrent network. As this 'jungle' of *axons* and *dendrites*, inter alia, features long-range connections, only little measured data exists regarding the connectivity. In order to close this gap of knowledge, neuroscientists have to try out different models. Auspicious approaches are, for instance, the concept of *liquid computing* [*Maass et al.*, 2002; *Jaeger*, 2001] or studies on *high-conductance states* [*Shelley et al.*, 2002; *Kumar et al.*, 2008]. To some extent, these investigations can be performed analytically, but the mixture of digital (spikes) and analog (neuron and synapse dynamics) data makes analytical calculations difficult.

Therefore, numerical simulations of neural networks are employed. There exist several approaches within the FACETS project with different focuses: Some simulation back-ends implement neuron and synapse models of high precision (e.g. the NEURON simulation software [*Hines and Carnevale*, 2003] implementing a Hodgkin-Huxley neuron model [*Hodgkin and Huxley*, 1952]), but accept either massive computational efforts (in case of software simulators, see e.g. [*EPFL and IBM*, 2008]) or a limited network size (in case of hardware implementations [*Renaud et al.*, 2007]). Other software simulators, like PCSIM [*Pecevski and Natschläger*, 2008] or NEST [*The NEural Simulation Technology Initiative*, 2008] aim at a faster computation using approximations of the above models: *"NEST is best suited for models that focus on the dynamics, size, and structure of neural systems rather than on the detailed morphological and biophysical*

properties of individual neurons."[4] But these concepts still apply exact parameters to the simulated components. The third approach within FACETS – the implementation of neural dynamics with analog VLSI[5] hardware – accepts a certain amount of loss of control over all parameters. In exchange, an inherent parallelism (close to the biological ideal) is gained, which allows for a fast emulation[6] that is almost independent of the network size.
A simple and certainly idealized framework for the application of simulators in neuroscience is shown in figure .1.
Ideally, any simulation back-end offers perfect control over all units that constitute the network, and yields perfect results in terms of the utilized models. But certain inhomogeneities are unavoidable and depend on the basic mode of operation of the back-end. For instance, computer simulators will always deal with inaccuracies due to discrete time steps. Particularly, when distributing a network simulation to a cluster, the principles of calculating all involved unit dynamics for the successive time steps can become challenging [*Morrison et al.*, 2005].
However, analog hardware suffers from inequalities of the physical implementation of similar units and of unintended interference of the units e.g. via the power supply.

In order to provide an interface to the different back-ends in abstract – biology inspired – terms, a unified language, called PyNN [*Davison et al.*, 2008, accepted for publication], was developed within the FACETS project. While a certain degree of abstraction will be found in any simulator interface, this common language provides additional advantages due the clean portability of experimental setups.
To overcome back-end specific differences it is feasible to apply generic universal network architectures, which are able to self-tune their properties to a certain amount, while providing the latitude to specialize functionality. This approach yields not only technical advantages, but is also biologically realistic, as all nerve cells are subject to noise and other perturbing influences: The most powerful network substrate known – the human cortex – exhibits such properties of universality and self-stabilization [*Baddeley et al.*, 1997].
Upon this foundation, the investigation of trained or self-learning networks can be assumed to lead to similar conclusions, independent of the utilized back-end.

As stated above, the minimization of inhomogeneities is a major challenge when developing an analog neural emulator. The current version of the FACETS Stage 1 Hardware – a prototype on the path to large integrations of analog neural circuitry on a silicon substrate – exhibits serious variations regarding the properties of neurons and synapses. In accordance with the framework presented in figure .1, this work addresses

- the investigation and maximization of homogeneity of the FACETS Stage 1 Hard-

[4] http://www.scholarpedia.org/article/NEST_(NEural_Simulation_Tool)
[5] VLSI: Very Large Scale Integration
[6] Since all units follow their respective inherent dynamics, the term *emulation* is used. In contrast, a *simulator* calculates the behavior of all units externally.

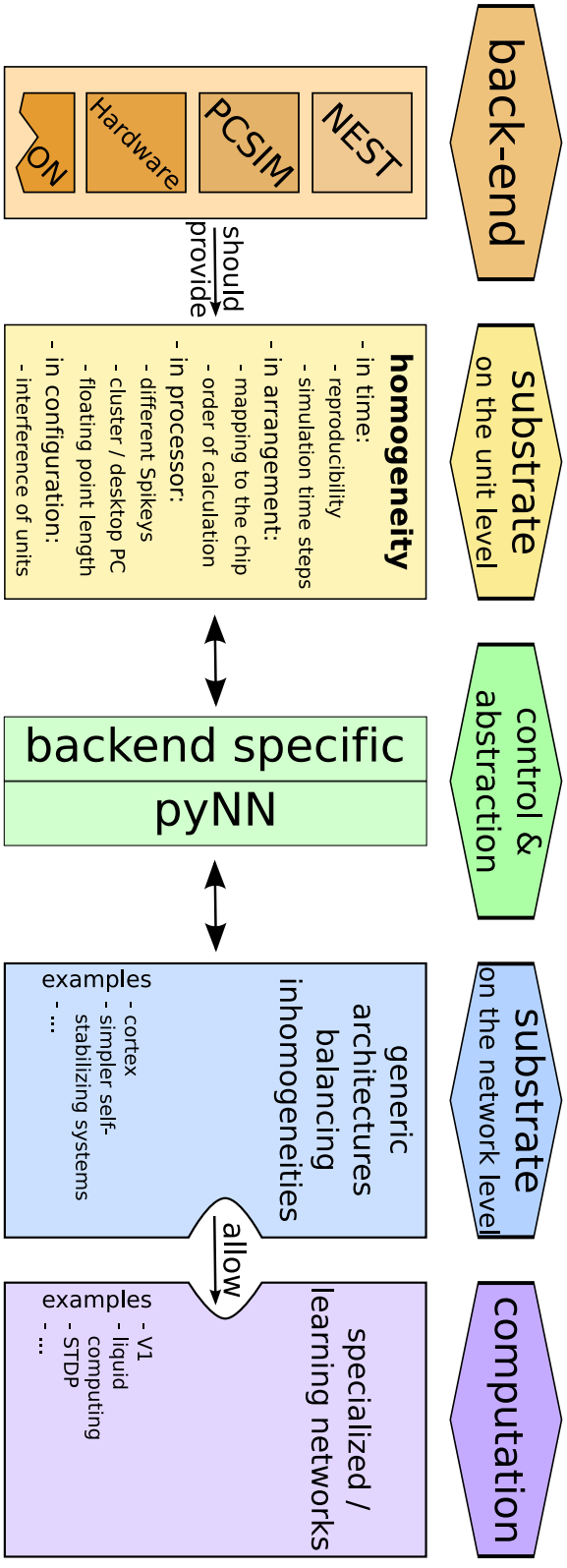


Figure .1: *A idealized framework regarding the application of neural simulators.*

ware,

- the proper mapping of abstractly defined networks to and operation of simulator back-ends,
- the application of self-stabilizing network architectures.

Basically, this thesis divides into two parts: The chapters I – III introduce the reader to the FACETS Stage 1 Hardware, the operating software package and utilized concepts on the network level, respectively. In chapter IV – VI, the author's contributions to the above issues are presented. In particular, a self-stabilizing network architecture is applied to the chip, in order to determine whether the degree of inhomogeneity, remaining after calibration and other efforts, can be reliably counterbalanced by a substrate on the network level.

I The FACETS Stage 1 Hardware

Rather than simulating neural networks in discrete time steps, the FACETS member UHEI[1] takes the approach to emulate neural and synaptic behavior via a CMOS[2] hardware device. Analog circuitry can reveal dynamics similar to common neuron and synapse models, enriching the spectrum of available simulation back-ends. Even though analog designs always deal with noise and variations due to the production process, these effects should not interfere with large scale networks, as long as the variances remain within a biologically realistic extent. On the other hand, the usage of integrated circuits allows highly accelerated emulations compared to the biological archetype since the time constants of all processes can be chosen extremely short. This speedup is almost independent of the network size, as all cells inherently operate in parallel.

The following sections give an overview of the current FACETS Hardware device (section I.1) and present the utilized neuron models (section I.2) and synapse models (section I.3) including their hardware implementation. Furthermore, details of the configuration and communication chain are addressed in brief (section I.4). Thereafter, restrictions of the FACETS Stage 1 Hardware on network topology and parameter configuration are discussed (section I.5). Finally, a glimpse into the future looks forward to the upcoming *Wafer-scale Integration* of the FACETS Hardware (section I.6).

As not stated otherwise, the following sections are based on [*Schemmel et al.*, 2007, 2006] and [*Grübl*, 2007] as well as personal communication with *Dr. Johannes Schemmel*[3], *Dr. Andreas Grübl*[4], *Daniel Brüderle*[5] and *Sebastian Millner*[6].

I.1 System Overview

The core of the FACETS Stage 1 Hardware holding the *analog neural network* is an ASIC[7] called *Spikey* which contains 384 neurons and almost 100.000 synapses per chip. It is currently run with a speedup of 10^5 compared to biology. In future, up to 16 chips can be interconnected, yielding networks of several thousand neurons and more than one million synapses. A micro photograph highlighting the elementary layout of the chip is

[1] Electronic Vision(s), Ruprecht-Karls-Universität Heidelberg
[2] CMOS: Complementary Metal Oxide Semiconductor
[3] schemmel@kip.uni-heidelberg.de
[4] agruebl@kip.uni-heidelberg.de
[5] bruederle@kip.uni-heidelberg.de
[6] sebastian.millner@kip.uni-heidelberg.de
[7] ASIC: application-specific integrated circuit

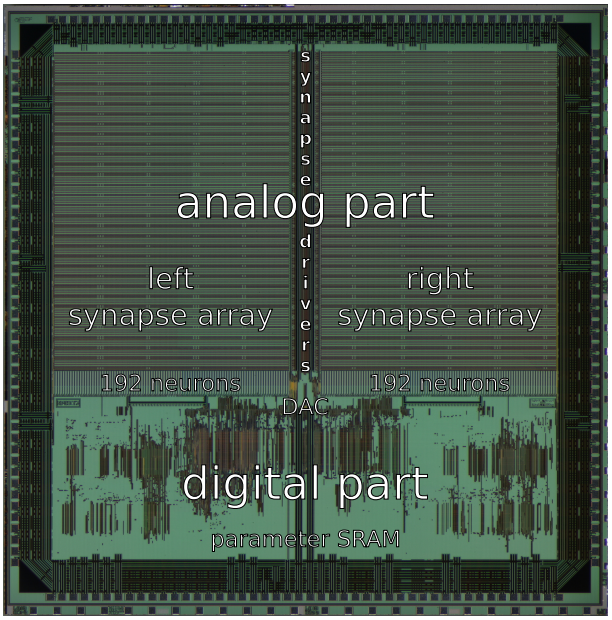


Figure I.1: *A micro photograph of the Spikey chip.*

shown in figure I.1.

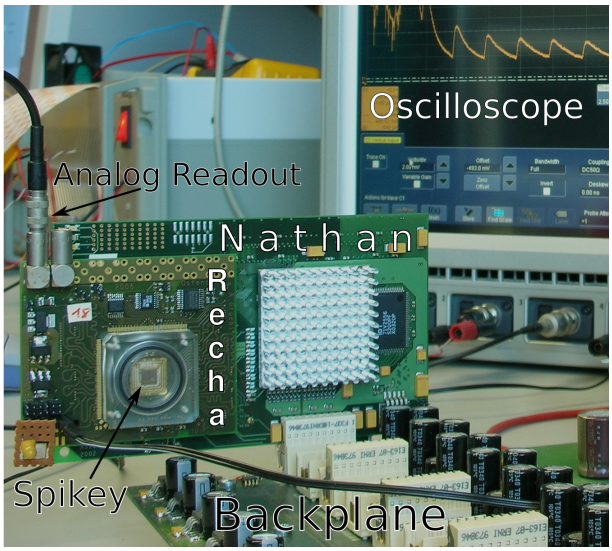


Figure I.2: *Experimental setup of the FACETS Stage 1 Hardware.*

In order to operate the chip, *Spikey* is bonded on an PBC[8] named *Recha*, which is plugged onto the so-called *Nathan*-board (see figure I.2). An *FPGA*[9] on the Nathan-board controls the configuration of and communication with Spikey. Up to 16 of such units can be operated by one *backplane*. Via a proprietary interface the backplane is connected to a Linux computer. Besides configuration and operation this setup allows digital readout of the action potentials generated by any neuron. Additionally, an oscilloscope provides access to the membrane potential in the sub-threshold regime via an *analog readout*.

The Spikey-chip measures $5 \times 5\,\mathrm{mm}^2$ and is fabricated by UMC[10] in a 180 nm CMOS process with one poly-silicon layer and six metal layers.

I.2 Neuron Model

The neurons emulated by the FACETS Stage 1 Hardware are conductance-based, leaky integrate-and-fire point-neurons [*Destexhe et al.*, 1998] obeying the following dynamics:

$$-C_m \frac{\partial V_m}{\partial t} = g_{leak} \cdot (V_m - V_{rest}) + \sum_k g_k(t) \cdot (V_m - V_{rev,exc}) + \sum_l g_l(t) \cdot (V_m - V_{rev,inh})$$

with $V_{rev,exc}$, $V_{rev,inh}$ and V_{rest} denoting three reversal potentials – excitatory, inhibitory and resting, respectively. The summation is done over all excitatory (k) and inhibitory (l) synapses. Their conductance course is presented in section I.3. The neurons have a membrane capacitance C_m and constantly (dis-)charge towards a resting potential V_{rest} via a conductance g_{leak}.
When a neuron's V_m exceeds a predefined threshold voltage V_{thresh}, it emits a digital action potential – in the following occasionally denoted as *spike*. At the same time its membrane potential is instantly set to a reset potential V_{reset}, where it is forced to remain for a refractory period τ_{refrac}.

I.2.1 Hardware Implementation

In the FACETS Stage 1 Hardware neuron membrane capacitances are realized using capacitors. Furthermore, each neuron contains a comparator circuit, that compares the membrane potential with the threshold. When $V_m > V_{thresh}$, the conductance between the capacitor and the reset potential is increased to a high value. A feedback circuit assures that this conductance rises quickly and does not return to zero once $V_m < V_{thresh}$. Additionally, the comparator circuit generates a square wave voltage that is transmitted to Spikey's digital part, where the spike is detected digitally and processed for feedback and readout.
The comparator behavior is adjusted via a bias current `icb`. This parameter also controls

[8] PBC: printed circuit board
[9] FPGA: Field Programmable Gate Array
[10] UMC: United Microelectronics Corporation, Taiwan

the period τ_{refrac} the capacitor remains connected to the reset potential. Furthermore, it determines the period of the square wave voltage sent to the digital part. Threshold, reset mechanism and `icb` are studied in detail in section IV.4.

I.3 Synapse model

The FACETS Stage 1 Hardware implements a double-exponential conductance-based synapse model. When triggered, the conductance between the neuron and the respective reversal potential jumps immediately to a starting value greater than zero, then it rises exponentially with time-constant τ_{rise} until it equals a previously defined threshold $w \cdot g_{max}$. Finally, it falls exponentially with time-constant τ_{fall} towards a very low level $w \cdot g_{rest}$.
For technical reasons each 192 synapses with the same presynaptic neuron share one parameter g_{max} (see section I.5). The individual synapse multiplies that value with a discrete weight-factor $w \in \{0, 1, 2, \ldots, 15\}$.
Furthermore, Spikey supports two different types of *synaptic plasticity*, namely *short term plasticity* (STP) and *spike time dependent plasticity* (STDP). While STP alters g_{max}, hence appearing as a pre-synaptic property, STDP changes the digital weight-factors w, thus describing characteristics of particular synapses.
In the hardware the information flow from a digital spike to an EPSP on a neuron's V_m is generated by a complex process: Whenever a neuron (or external input) emits a spike, the action potential is transferred digitally to the assigned synapse driver (see also section I.5), which is triggered to generate a linear voltage ramp. In the synapse this voltage ramp is transformed to an exponential current ramp, controlling the conductance between the post-synaptic neuron and the respective reversal potential. The units involved in this process will be described in the following.

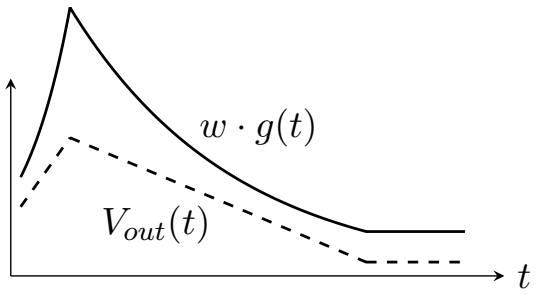


Figure I.3: *Time course of Spikey's double-exponential conductance-based synapse model. The synaptic conductance $w \cdot g(t)$ is shown alongside the linear voltage ramp $V_{out}(t)$ generated by the synapse driver (in arbitrary units).*

I.3.1 The Synapse Driver

Synapse drivers receive action potentials from both Spikey's digital part and the neuron blocks as feedback connections and transform them into linear voltage ramps, for transmission to the synapses. Furthermore, they implement *short term plasticity*. Thus, the synapse drivers control the dynamics of the synapse model. This paragraph outlines their basic operation (for details see [*Schemmel et al.*, 2007]). Figure I.4 shows the synapse driver circuit.

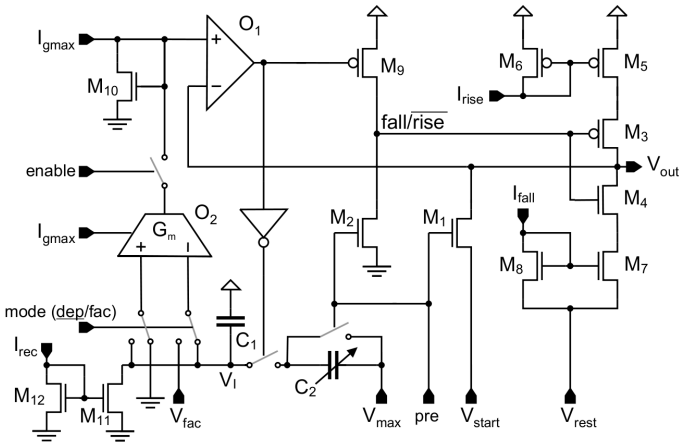


Figure I.4: *Synapse driver circuit implementing short term plasticity. The STP mechanism is located in the lower left part of the circuit.*

A digital AP enters the circuit as a square wave voltage (`pre`-signal) generated by the digital part. The signal can either originate from an external input, a specified neuron in the same half of the chip or a specified neuron in the other half (see section I.5).
The output voltage transmitted to the synapses is called V_{out}. When a pulse enters the driver, V_{out} is set to V_{start} via M_1. At the same time the `fall/rise`-line is pulled down, initiating the rising voltage ramp: V_{out} is increased linearly in time via the current mirror M_5/M_6, while the connection M_4 is interrupted. The slope of the voltage ramp is controlled via I_{rise}. As V_{out} exceeds the driver strength determined by I_{gmax}, the comparator O_1 pulls up the `fall/rise`-line[11]: V_{out} begins to fall linearly towards V_{rest} with a slope, defined by I_{fall}. The time course of V_{out} and the resulting synaptic conductance $w \cdot g(t)$ are drafted in figure I.3.
The circuitry realizing STP is located in the lower left part of the synapse driver. Its operation is presented in a separate paragraph.
Besides the control voltages and currents, each synapse driver has a *configuration byte*. Its structure is shown in table I.1. Furthermore, two synapse rows can be combined in order to increase the weight resolution. This feature is not yet supported by the software

[11] The transformation of I_{gmax} into a voltage for comparison to V_{out} is performed via an 'inverse synapse circuit'. Thus, the actual synaptic strength is proportional to the control current: $g_{max} \propto I_{gmax}$.

and will not be presented in this work.

Bit	Description
0	`0`: external input, `1`: feedback connection
1	selected feedback source: `0`: adjacent block, `1`: this block
2	`1`: set row to be excitatory
3	`1`: set row to be inhibitory
4	`1`: enable the STP-mechanism
5	select STP mode: `0`: facilitation, `1`: depression
7 6	size of capacitor $C_2 = \frac{2X+1}{8} \cdot C_1$ ($X \in \{00, 01, 10, 11\}$)

Table I.1: *The structure of the synapse driver configuration byte.*

I.3.2 Synapses and Reversal Potentials

The synapses are arranged on two two-dimensional grids, so-called *synapse arrays*, of size 256×192. Each *synapse driver* operates a *row* of *192 synapses*. Each set of 192 synapses shares the configuration bit, whether all synapses are *excitatory* or *inhibitory* (see table I.1). Besides to a driver, each synapse is also assigned to a post-synaptic neuron forming 2×192 *columns*.
A synapse transforms the driver's linear voltage ramp V_{out} into an exponential current $I_{out} \propto w \cdot \exp(\rho \cdot V_{out})$, scaling with an individually adjustable, discrete weight-factor $w \in \{0, 1, 2, \ldots, 15\}$ and generating the designated exponential behavior.[12] I_{out} can either be connected to an 'excitatory' or an 'inhibitory' wire. Both head towards the post-synaptic neuron and are shared by 256 synapses operated by different synapse drivers. Thus, all excitatory currents $\sum_{0<i<255\,;\,i\,\text{exc.}} I_{out,i}$ sum up to a total excitatory current I_{exc}. In the same way a total inhibitory current I_{inh} sums up.
I_{exc} and I_{inh} control the conductances between the neuron's capacitor and the respective reversal potentials $V_{rev,exc}$ and $V_{rev,inh}$. In the following this connection is denoted as *ion channel*, according to its biological counterpart.

I.3.3 Short Term Plasticity (STP)

When firing several times in succession, biological synapses will change their effective weight over time spans of some milliseconds to seconds. This effect is known as *short term depression* and *facilitation*. The short term plasticity mechanism implemented in Spikey is motivated by a phenomenological model developed by *Markram et al.* [1998].

[12] The value of the constant ρ is of no importance, since V_{out} can be adjusted in a wide range.

Hardware Model

Every *synapse driver* of the FACETS Stage 1 Hardware supports two modes of *short term plasticity*, namely *depression* and *facilitation*. The basic idea is to introduce a time varying *inactive partition* I with $0 \leq I \leq 1$. It decays exponentially with time constant t_{rec}, while every AP processed by the synapse driver increases I by a fixed fraction towards the maximum. This idea leads to the following dynamics for the inactive partition:

$$\dot{I} = -I/t_{rec} + C \cdot (1 - I) \cdot \delta(t - t_{AP})$$

with $0 < C < 1$ and t_{rec} denoting adjustable constants. The impact of the inactive partition on the effective synapse weight is controlled via a scaling factor λ.
In **depression mode** the inactive partition reduces the synapse's effective weight, thus:

$$w_{dep} \propto 1 - \lambda \cdot I$$

In **facilitation mode** the inactive partition is added to the static synaptic efficacy: $w_{fac} \propto 1 + \lambda' \cdot I$. For practical reasons (see below) it is useful to keep the same scaling factor λ but introduce another constant N, and write the impact of I in the form

$$w_{fac} \propto 1 + \lambda \cdot (I - N)$$

Hardware Implementation

The model presented above results in additional circuitry in the synapse driver (see fig. I.4). This paragraph outlines the basic operation of the STP-mechanism. For details see [*Schemmel et al.*, 2007]. The influence of dynamic synapses on the membrane potential is shown in figure I.5.
Short term synaptic plasticity can be switched on via the `enable`-signal. The `mode`-signal selects the type of STP. The inactive partition I corresponds to the voltage V_I.[13] The current I_{rec} adjusts the decay time constant t_{rec}. When triggered by a spike, charge is transferred from C_1 to the adjustable capacitor C_2[14]. This leads to a rapid change of V_I:

$$\Delta V_I = \frac{C_2}{C_1 + C_2} \cdot (V_{max} - V_I)$$

Particularly, V_I never exceeds V_{max}, which controls the scaling parameter λ:
The output of the operational transconductance amplifier (OTA) O_2 adds a (positive or negative) current I_{stp} to the static synaptic weight I_{gmax}. This current is proportional to the voltage $\hat{V}$ between the two inputs of the OTA. Since O_2 is biased with I_{gmax}, its output is also proportional to this current. Hence, $I_{stp} = \mu \cdot I_{gmax} \cdot \hat{V}$ with a constant μ. This leads to:

$$I_{total} = I_{gmax} + I_{stp} = I_{gmax} \cdot (1 + \mu \cdot \hat{V})$$

[13] More precisely, we identify $I = V_I/V_{max}$, as I is normalized to 1, while V_I is normalized to V_{max}.
[14] C_2 can be set to $i \cdot C_1/8$, with $i \in \{1, 3, 5, 7\}$.

In case of **depression** $\hat{V} = -V_I$, and thus, $\lambda = \mu \cdot V_{max}$.
In case of **facilitation** $\hat{V} = V_I - V_{fac}$. The adjustable reference voltage V_{fac} is introduced to increase control over the weight-range covered by facilitation. This becomes necessary, since the OTA cannot supply higher currents than its bias. For $N = V_{fac}/V_{max}$ it is $\lambda = \mu \cdot V_{max}$.

`enable`, `mode` and C_2 are set via the synapse driver *configuration byte* listed in table I.1. The voltages and currents are generated by the on-chip DAC (see section I.4).

All hardware parameters are summed up and compared to the variables of the PCSIM implementation of the *FACETS Hardware Synapse* in table V.1.

I.3.4 Spike Time Dependent Plasticity (STDP)

Besides *short term plasticity*, that does not alter the synaptic strength permanently (it recovers with time-constant t_{rec}), the FACETS Stage 1 Hardware provides a persistent plasticity mechanism on long time scales – namely, *Spike Time Dependent Plasticity* (STDP). The model implemented in Spikey is presented in [*Schemmel et al.*, 2007, 2006] and is based on [*Song et al.*, 2000; *Bi and Poo*, 1997].
Basically, for each synapse the time points of pre- and post-synaptic action potentials are compared. The resulting $\Delta t = t_{pre} - t_{post}$ determines, whether the APs occurred *causal* ($\Delta t < 0$) or *acausal* ($\Delta t > 0$). Causal spike pairs strengthen the synapse (w is slightly increased), acausal spike pairs weaken it (w is slightly decreased).
Since Spikey's weight matrix only supports discrete weights, the impact of numerous *causal* and *acausal* spike pairs is stored temporarily, until a preset threshold is exceeded. Then a *STDP controller* within the digital part updates the respective discrete weight w.

I.4 Configuration and Readout

All parameters operating the FACETS Stage 1 Hardware and describing a neural network experiment (namely, configuration and spike input) must be transfered to the Chip. Spikes, emitted by the neurons, as well as analog membrane potentials must be read back. This section briefly addresses the components involved in this *read-* and *write-process* on the hardware level. The software layers are presented in chapter II.

I.4.1 The Parameter RAM

Most analog parameters are provided as 10-bit digital numbers in units of $\frac{\texttt{irefdac}}{10 \cdot 1023}$. `irefdac` is a reference current provided externally by the *Recha*-board. A *digital-to-analog converter* (DAC) within Spikey generates the desired currents. These currents are stored within *current memories*. Also voltages (so called `vouts`) are generated using currents from the DAC, by creating them via a resistor and storing them on capacitors.
Since both currents and voltages cannot be stored in the analog domain for more than

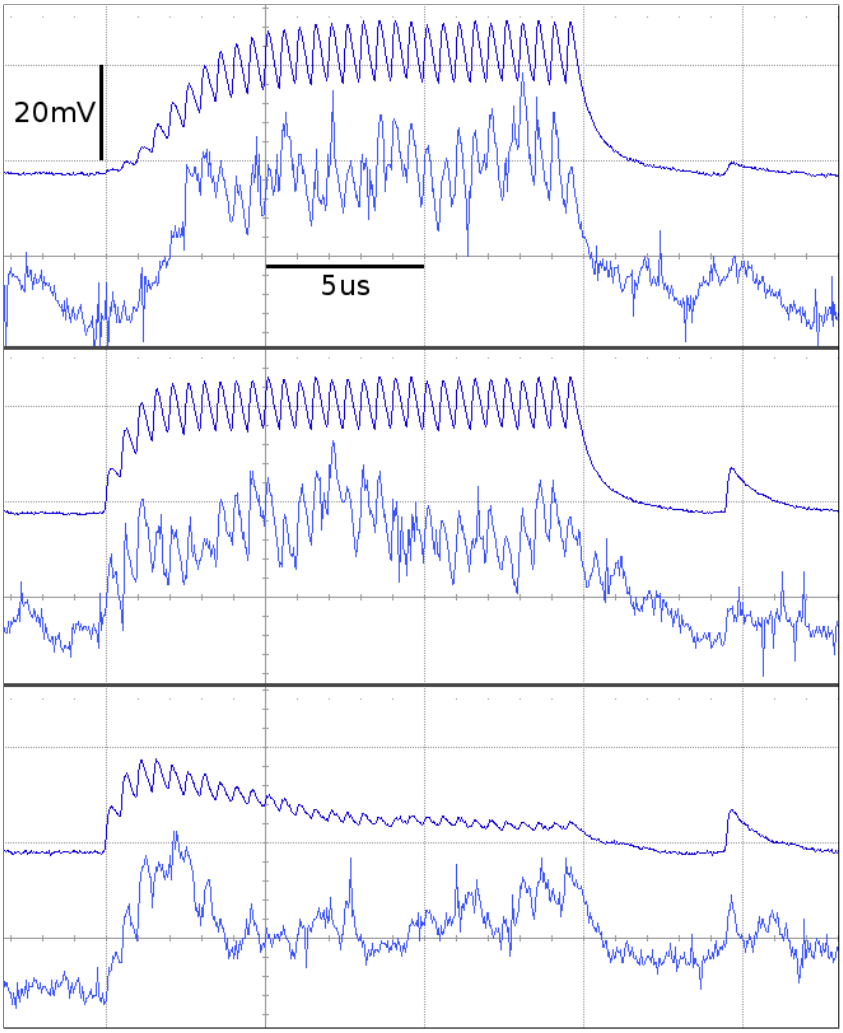


Figure I.5: *Comparison of static and dynamic synapses on the FACETS Stage 1 Hardware. A neuron was stimulated with Poisson background noise (different in every trial). Additionally, one synapse driver, injecting equidistant spikes with 20 Hz biological frequency (`speedup` $= 10^5$) and one spike appended after 500 ms, was configured to be either **facilitating (top)**, **static (middle)** or **depressing (bottom)**. The figures show the membrane voltage trace as seen on the oscilloscope. Thus, time and voltages are given in hardware units. While each lower trace displays a single trial (with the Poisson input clearly visible), for the upper traces 100 trials were averaged revealing the influence of the specified synapse driver. The three plots only differ in the `mode` of this driver. Obviously, depressing synapses become weaker the more APs being processed in succession, which can be seen in the decreasing height of the singe PSPs. After 500 ms without activity the PSP induced by the presynaptic spike has almost the same height as the first induced PSP, which indicates that the inactive partition has almost recovered to its initial value. In contrast to static synapses a facilitating driver develops its full strength only after a couple of transmitted APs, since its starting strength is smaller by the factor $(1 - \lambda \cdot N)$. The configuration of the synapse drivers equals the setup used on a network level as presented in section VI.3. Both workarounds discussed in section IV.5 have been applied to the short term plasticity mechanism. The following parameters were chosen: $V_{stdf} = 0$ (minimum `vout`), V_{fac} bypassed, `VSTART`$=0.25\,\mathrm{V}$, $V_{dtc} = 0.7\,\mu\mathrm{A}$, $C_{2,fac} = \frac{3}{8}C_1$ and $C_{2,dep} = \frac{1}{8}C_1$.*

a couple of milliseconds, they have to be rewritten by the DAC periodically. For that purpose they are stored in a SRAM module in Spikey's digital part. The DAC updates all values (about 3000 in total) one after another. Then it starts again with the first address. A full parameter-RAM refresh cycle currently takes about 15ms.

I.4.2 External Voltages

Some voltages are generated externally by a DAC on the *Recha*-board. Examples are the on-chip DAC reference current `irefdac` and some more specific, global voltages like V_{start} and V_{rest} of the *synapse driver voltage ramp*. Unlike voltages generated inside the Chip (see also section I.5), external voltages are subject to no relevant restrictions regarding their adjustability. But it was found that these voltages can become unstable in case of high load. As this defect was discovered in November 2008, its cause was not completely understood when this thesis was written.

I.4.3 Spike Readout and Spike Input

When a neuron emits a spike, the square wave voltage, indicating the spike (see section I.2), is processed by a *priority encoder*, detecting the *neuron address*, and a *time-to-digital converter* (TDC), generating a *sub-time stamp*. This information is passed to an *output buffer*, that adds a *system-time stamp*. Each 64 neurons share one unit consisting of priority encoder, TDC and output buffer. In total, Spikey contains six of such units for spike readout. All buffers share one bus to transfer their data to the *Nathan-FPGA*, which stores the spike data in an ordinary RAM. This output spike data is read back, translated from hardware time to biological time by the software and, eventually, provided to the user.
The current version of the FACETS Stage 1 Hardware (Spikey III) reveals errors, when multiple neurons of the same block of 64 neurons are recorded and emit spikes simultaneously: The priority encoder can enter a locking state and stops further spike output. Only the last 5 - 9 neurons of each block can be recorded reliably at the same time.
Furthermore, Spikey's output rate is limited by design: Since all output buffers share one bus, buffer overflows can occur due to this bottleneck. Typically, for larger networks a total readout spike rate of about 2000Hz (in biological time) can be processed without buffer overflows.[15]

The external spike input works likewise: For each spike the software generates *system-time stamp*, *sub-time stamp* and *synapse driver address*. This information is written into a *playback memory* on the Nathan board. Via the Nathan-FPGA and a bus system this data is passed to an *input buffer* on Spikey. At the proper system-time, sub-time and driver address are transmitted from the buffer to a *digital-to-time converter* (DTC) which generates the `pre`-signal for the respective synapse driver at the precise time point, determined by the *sub-time stamp*.

[15] The rate in biological time depends on (a) the system clock frequency and (b) the hardware-to-biology speedup. Currently, $\nu = 100\,\mathrm{MHz}$ and $\texttt{speedup} = 10^5$.

Each 64 synapse drivers share one unit consisting of input buffer and DTC, yielding eight of such input units per chip.

I.4.4 Analog Readout and Oscilloscope

The FACETS Stage 1 Hardware supports monitoring of several internal voltages via an analog output, called `ibtest` pin. The pin can be connected to an oscilloscope, that provides a convenient readout via a TCP/IP port.
Any neuron's membrane potential and any `vout` voltage can be viewed via the `ibtest` pin. Normally, only one parameter can be seen on `ibtest` at a time.[16]

I.5 Network Topology and Parameter Constraints

Although most parameters are adjustable and any two neurons can be connected synaptically, the FACETS Stage 1 Hardware reveals several constraints when mapping larger networks. These constraints span the values a parameter can take, configurations multiple units share and network topologies impossible to map.
Some constraints are acceptable and intended by design. Others originate from faults of particular units, that cannot be counteracted during the calibration process outlined in section II.2.

I.5.1 Topology

When a network shall be emulated by the Hardware, one has to decide, which hardware neuron to map every 'biological' neuron to. This yields the question, if all desired synapses can be established. This mapping process can become a challenging task for large networks. One approach regarding mapping algorithms is presented in section II.3. This paragraph addresses the underlying hardware topology.

Restrictions on Connectivity

Each synapse array is operated by 256 synapse drivers, 64 of which are reserved for external input only. For driver 0 to 191 there exist three choices[17] regarding the source of the `pre`-signal. The source is set via the *configuration byte* (see table I.1). For synapse driver $n \in \{0, \ldots, 191\}$ there exist the following sources:

- external input
- APs of the nth neuron in the same block
- APs of the nth neuron in the other block

[16] Actually, Spikey allows simultaneous analog readout of up to eight (almost arbitrary) neurons' membrane potentials. Since this feature is used rarely and uses additional pins on the *Recha* board, it will not be discussed, here. Additionally, a workaround providing *write-access* to multiple `vouts` simultaneously under certain conditions is presented in section IV.5.

[17] ignoring merging of adjacent synapse drivers

The selected source is valid for all synapses operated by the same driver. Hence, it is impossible, that two neurons with the same index '*n modulo 192*' serve as pre-synaptic neurons for *any two neurons located in the same half of the chip.*

Restrictions on Synapse Configuration

Furthermore, all synapses operated by the same synapse driver share its *configuration byte*. Especially, all these synapses are either excitatory or inhibitory – a condition, that is in accord with the common network models. A real constraint is the shared configuration of *short term synapse dynamics* (see table I.3): This limits the network size to 192 neurons for the setup presented in section VI.3.

I.5.2 Parameter Ranges

All parameters can only be set within a certain range. As stated in section I.4, external voltages can be set to any value needed. But most voltages and currents are generated by the on-chip DAC (see section I.4). Such currents can only be set from 0 to 2.5 μA[18] with a 10-bit resolution, yielding a step-size of 2.44 nA.
More restricted are the internally generated voltages `vouts`: The current revision (Spikey III) allows for a `vout`-range from 0.6 V to 1.6 V. The exact interval is determined by the individual `vout`. Especially, the lower bound prevents reasonable STP-parameters as revealed in section IV.5. A workaround to lower a small number of `vouts` is presented there, too.

As described in section I.3 all synapses, operated by the same synapse driver, share their maximum synaptic weight due to I_{gmax}. As the synaptic weight-factor w is a 4-bit digital value, the weakest (non zero) synapse in a row must at least have $w_{weak} = \frac{1}{15} \cdot w_{strong}$, for the strongest synapse being of weight w_{strong}.

I.5.3 Shared and Individual Parameters

As stated above, all analog circuitry is subject to variations due to the production process. Many resulting effects on neural and synaptic behavior can, in principle, be counteracted by readjusting the respective parameters. This calibration process is hindered, since many voltages and currents are shared by multiple neurons or synapse drivers. Hence, the universality of a parameter can become crucial for the success of a calibration algorithm.

Table I.2 and table I.3 list parameters, which are commonly utilized in order to set up Spikey properly, and show whether a parameter can be set individually for a unit or is assigned to multiple, similar parts. *'Global'* indicates that a voltage or current supplies the entire chip. A *'shared'* parameter serves all units that are located in the same half of the chip AND have an index of the same parity (odd or even) – resulting

[18] Currently, it is `irefdac` = 25.0 μA. A parameter that is not expected to be changed.

in four independent equivalent voltages/currents supplying one parameter to one chip. *'Individual'* parameters refer to exactly one unit, meaning that there exist as many independent voltages/currents as neurons or synapse drivers. In case of `vouts`, every `vout` voltage includes a bias current `voutbias` of the same universality which is omitted in the listing. Some parameters have different names in the hardware design and the software environment and are denoted with both names, here.

parameters referring to neurons

parameter	**universality**
I_{leak} (= g_leak)	individual
icb	individual
all reversal potentials (exc, inh, resting)	shared
reset potential	shared
threshold voltage	shared

Table I.2: *Universality of parameters referring to neurons. A* 'shared' *value involves 4 voltages/currents with each entity supplying 96 neurons.*

parameters referring to synapse drivers

parameter	**universality**
drviout (= I_{gmax})	individual
adjdel	individual
drvifall (= I_{fall})	individual
drvirise (= I_{rise})	individual
V_{dtc} (= I_{rec})	shared
V_{fac}	shared
V_{stdf} (= V_{max})	shared
vcb	shared
vplb	shared
V_{rest} (= VREST)	global
V_{start} (= VSTART)	global

Table I.3: *Universality of parameters referring to synapse drivers. A* 'shared' *value involves 4 voltages/currents with each entity supplying 128 synapse rows.*

I.6 Stage 2 – Wafer-Scale Integration

The FACETS Stage 1 Hardware is a *chip based system*: Each identical unit – the Spikey chip – needs an extensive set of other devices (RECHA, NATHAN and Backplane) in

order to communicate with other units, i.e. to synaptically connect neurons on different chips, involving a minimum spatial distance between the units. Considering signal propagation delays, this concept inherently limits the number of interconnectable neurons. Hence, a system being capable of emulating large neural networks needs to feature a high integration density.

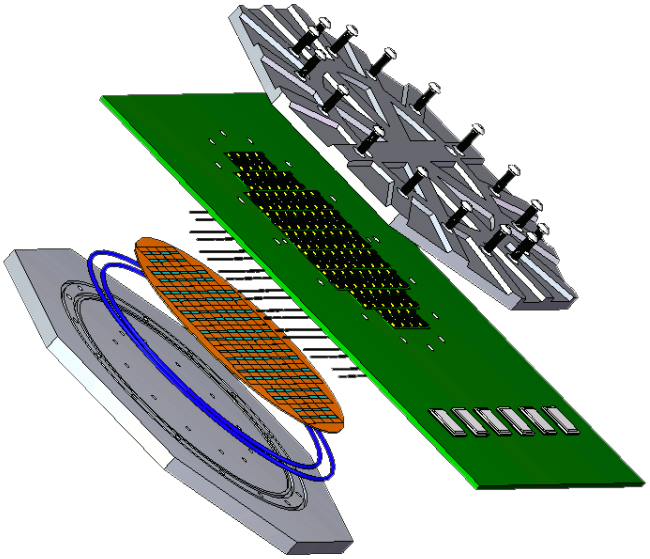


Figure I.6: *Schematic of the FACETS Stage 2 Hardware as presented in [Schemmel et al., 2008]. Between the mounting brackets are the wafer and the motherboard, linked with elastomeric connectors.*

Rather than cutting the identical units into dies, the upcoming *FACETS Stage 2 Hardware* aims for a *wafer-scale integration* meaning that the ASICs holding *analog neural network cores* are interconnected on the uncut wafer during a post-processing step (see figure I.6). For operation a motherboard is attached 'on top' of the wafer using elastomeric connectors. A wafer of 20 cm diameter is supposed to contain about 350 *HICANN*[19] building blocks. As each HICANN implements 512 neurons and 512×256 synapses[20] one wafer can theoretically emulate neural networks consisting of more than 180 thousand neurons and 45 million synapses – achieving a new dimension of integration density. For an expected speedup factor of 10^4 and an average pre-synaptic firing rate of 10 Hz this yields an event rate of 12 Giga-events per sec and HICANN requiring a vast communication bandwidth. As each event involves charging and discharging of the wire capacitances a new low-voltage signaling scheme was developed in order to limit the power consumption.

[19] HICANN: High Input Count Analog Neural Network

[20] Optionally, multiple neurons can be combined to bigger neurons supporting up to 16k pre-synaptic inputs per neuron.

The FACETS Stage 2 Hardware supports STDP and short-term plasticity. Furthermore, it extends Spikey's conductance-based leaky integrate-and-fire neuron model to the aEIF-model[21] proposed by [*Brette and Gerstner*, 2005]. This neuron model inherently minimizes the impact of threshold variations (like those discussed in section IV.4) on the firing behaviour.

The tape-out of the first HICANN prototype is planned for early 2009. Further details on neuron and synapse models and the technical realization of the FACETS Stage 2 Hardware are found in [*Schemmel et al.*, 2008]. Network topology and the mapping of biologically relevant network models on the future system are addressed in [*Fieres et al.*, 2008].

[21] aEIF: adaptive exponential integrate-and-fire

II Mapping and Communication Framework

As described in chapter I, almost all parameters, characterizing the emulated network in the FACETS Stage 1 Hardware, are represented by voltages and currents. In order to use the chip as a neural simulator, an abstractly described neural network must be mapped to the hardware resources, and all biological values have to be expressed in terms defined by the circuity. Additionally, components not belonging to the actual network – like Spikey's digital part or the entire communication chain – need to be configured and operated.
Furthermore, any approach for operating the hardware will aim for an abstract framework that hides these technical details from the user. Ideally, the experimenter is allowed to express his thoughts in a language close to the biological model and independent of the utilized simulation back-end.

Section II.1 addresses the multilayer software package, which performs the above tasks and was developed by the Electronic Vision(s) group. In section II.2, a closer look is taken at the parameter translation, which also considers variations of the individual hardware components due to the production process. Finally, in section II.3, a concept is described, on how a network can be mapped to the chip efficiently.

II.1 Software Structure

The software operating the FACETS Stage 1 Hardware can basically be divided into three layers. From the experimenter's view, the network, the execution of the emulation and the readout are described in the *python-based language PyNN*. Through the *Hardware Abstraction Layer* all commands and data are passed to the *hardware specific low level code*, which carries out the communication with the chip.
While the data structures used in the top layer reflect the biological framework, the low level code rather images the chip layout. Figure II.1 gives an overview of the software layers, which are described in the following.

II.1.1 PyNN

For comparison of different simulation back-ends and the verification of simulation results, it is beneficial to describe all experiments in a unified, portable language. The *PyNN-project*[1] [*Neural Ensemble*, 2008] aims for the development of such a common

[1] PyNN: Python package for simulator-independent specification of Neuronal Network models, see: http://neuralensemble.org/trac/PyNN

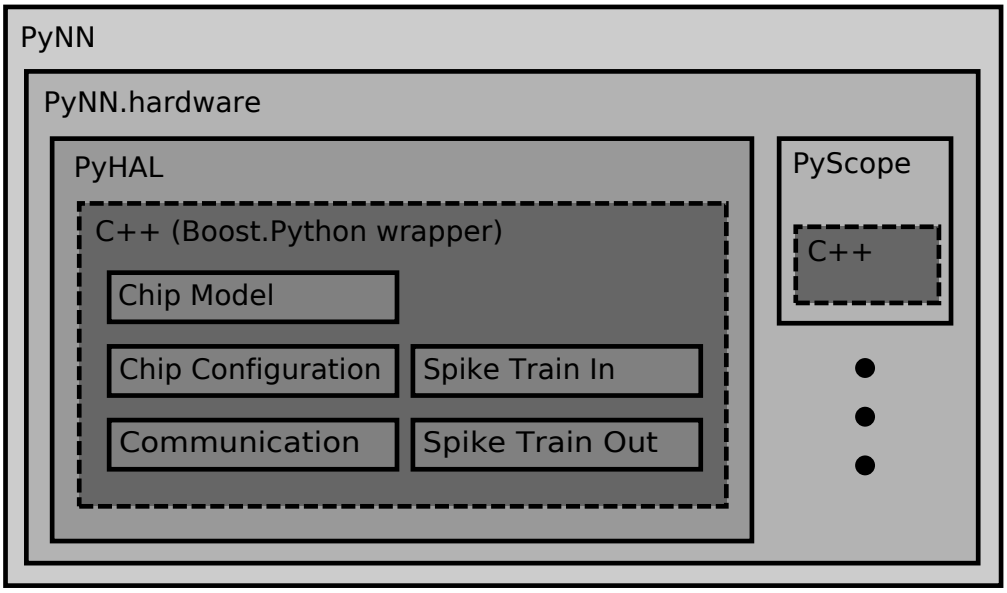


Figure II.1: *Software overview showing the software structure of the FACETS Stage 1 hardware system. With friendly permission of Eric Müller [Müller, 2008].*

API.[2] PyNN is developed within the FACETS project and is currently supported by NEST [*Gewaltig and Diesmann*, 2008], PCSIM [*Pecevski and Natschläger*, 2008], NEURON [*Hines et al.*, 2008] and the FACETS Stage 1 Hardware.[3]
The PyNN definitions include functions for setting up networks (e.g. `create()` or `connect()`), classes for generating stimuli (e.g. `SpikeSourcePoisson`) and commands to acquire the simulation results (e.g. `record()`). Besides these procedural commands, the API defines two objects for the management of sets of similar neurons (`Population`) and synaptic connections between those (`Projection`).
Since each simulator implements the same API, it suffices to change a single line of code to switch to another back-end:

```
import pyNN.pcsim as pyNN ⟶ import pyNN.nest2 as pyNN
```

The functions implemented in the `pyNN.hardware.stage1` module communicate with the lower software layers. The `PyScope` module provides access to the oscilloscope. The `PyHAL` module is presented in detail.

II.1.2 Hardware Abstraction Layer

The storage and organization of the commands and network configuration received from the *PyNN-layer* is performed by the so-called Hardware Abstraction Layer [*Brüderle*

[2] API: Application Programming Interface
[3] The `pyNN.hardware.stage1` module is not included in the release of the PyNN package, since it cannot be used without the hardware.

et al., 2007]. As written in Python, it is abbreviated as *PyHAL*. The major functionality of this layer is organized in two submodules, namely `config` and `buildingblocks`.

The `buildingblocks` contain a `Network` class, which holds the `Neurons` and manages some common quantities (e.g. the network size or the number of external inputs). The connectivity is purely stored by the `Neuron` objects, which keep track of all incoming and outgoing `Synapses`. Besides the synaptic weight, each `Synapse` object is able to store its *STP*- and *STDP*-configuration.
External stimuli are not contained in the `Network`, but processed directly by the `HWAccess` object provided in the `config` module. Its main tasks are to

- assign the `Neurons` to the hardware neurons,
- load the calibration data and translate all biological values to hardware parameters (see section II.2),
- pass the external stimuli to the low level software,
- initialize the configuration of the chip and
- read back the emulation results (spikes only).

Most of the such prepared configuration is stored in a `SpikeyConfig` object. Via the *Boost.Python*[4] wrapper, the communication to the low level software is established.

II.1.3 Low level Software

The low level, hardware specific code is written in *C++*. It implements an image of the structure of the actual hardware, e.g. the `Spikey` object. The low level software provides functions to send all configuration data and the spike trains to the chip, and to read back the spike output.

II.2 Mapping Biological Networks to the Hardware

In contrast to a software simulator, which can be designed to use a native data structure, the parameters, needed to operate electrical circuitry, clearly differ from biological terms. Furthermore, since permutations applied to the mapping of abstractly defined neurons to the hardware neurons should influence the emulation results as little as possible, an individual adjustment of the values, that are written to the particular components, is suggested. Table I.2 and table I.2 in section I.5 give an overview of parameters, which can be set for each neuron or synapse driver, individually. In the following, the basic concepts of the existing calibration algorithms are outlined. The interpretation of many parameters depends on the specified `speedup` factor, which relates the 'emulation time' to the real 'hardware time'. As the calibrations base on each other, it is recommended to keep the order of the description.

[4] http://www.boost.org/doc/libs/1_34_0/libs/python/doc/index.html

vout calibration: This is an actual hardware calibration, in a more restricted sense. Since all `vout` voltages can be measured via the `ibtest` output (see section I.4), a rather precise mapping between the written parameters and the resulting voltages can be determined. When the correct hardware voltages are known, the range of biological voltages (typically, $-80\,\mathrm{mV}$ to $-55\,\mathrm{mV}$, ignoring the excitatory reversal potential) can be mapped to the hardware voltages (currently, about $0.6\,\mathrm{V}$ to $1.6\,\mathrm{V}$ – cf. section I.5).

output pin calibration: As the `vout` voltages were measured internally, they can be used to calibrate the analog output pins, which are used to measure neuron membrane voltage traces on the oscilloscope. For each pin, a neurons resting potential V_{rest} is set to specified voltages. The values seen on the oscilloscope allow for an adjustment of the analog readout units.

icb calibration: A calibration algorithm, addressing variations in the threshold voltage and the reset mechanism. This phenomenon is studied in detail in section IV.4. Also, approaches for moderating this issue are presented there.

$\tau_{\mathbf{mem}}$ calibration: The current $\mathtt{I_{leak}}$ controls the conductance g_{leak} between the membrane and the resting potential V_{rest}. It is, for both biology and electronics, $\tau_{mem} = C_m/g_{leak}$ (in their respective units). The membrane time constant can be determined by setting $V_{rest} > V_{thresh}$. Thus, the neuron will spike periodically. The time constant of the exponential 'decay' can be measured either via the firing rate or directly via the trace of the membrane voltage. An individual adjustment of $\mathtt{I_{leak}}$ to the desired τ_{mem} for every neuron additionally considers variations in both the conductance $g_{leak}(\mathtt{I_{leak}})$ and the membrane capacitance C_m.

drviout and drvifall calibration: In order to determine the synaptic efficacy of a synapse driver, an input is connected to a neuron with a fixed discrete hardware weight w. The amplitude and the time course of the EPSPs allow the adjustment of the maximum synaptic conductance $g_{max}(\mathtt{drviout})$ and the time constant of the falling edge $\tau_{fall}(\mathtt{drvifall})$, e.g. by comparison to software simulations. In addition, variations of the transistors M_7, M_8, M_{10} and the comparator O_1 in the synapse driver circuitry (shown in figure I.4) are compensated. The synaptic efficacy is subject to various interferences. Thus, the described method just outlines the basic idea. Some sources of error are discussed in section IV.1 and section IV.3. A detailed description of a calibration routine is presented in section IV.2.

weight transformation calibration: To consider variations in the membrane capacitance C_m and the ion channels (see section I.3) of the neurons, the weight vector of incoming connections can be adjusted for each neuron. Due to the discrete nature of the hardware weights, this results in a loss of control over the exact configuration of the individual synapse. For that reason, this method should only be applied to larger

networks, when control over the average synaptic weight is more important than the individual synapse. Possible measures for calibration are mean firing rates under specified stimuli or the amplitude of individual EPSPs.

II.3 Graph Model

As mentioned above, the software has to decide, to which hardware neurons map the neurons defined in PyNN. This *mapping process* has not only to consider the hardware topology of possible synaptic connections, but also parameters shared by multiple units (see section I.5) and communication bottlenecks due to limited bandwidth.
For the smaller *Stage 1* hardware, this task could be accomplished by rather straight algorithms – or simply by the experimenter. But the *Stage 2 wafer-scale integration*, presented in section I.6, exhibits a much more complex topology caused by different hierarchical levels and multiple communication layers. Presumably, many larger networks will not be perfectly mappable to the hardware. So, it must be decided, which of the available (imperfect) mappings is to be chosen. Preferably, this evaluation does not increase the configuration overhead noticeably.

The FACETS members Dresden[5] and Heidelberg[6] addressed this issue systematically and developed a generic framework, called the *Graph Model*. Basically, – before the actual mapping process – models of both the 'biological' network (*bio graph*) and the hardware substrate (*hardware graph*) are generated. The graphs consist of several building blocks: *Nodes*, for instance, represent neurons, parameter sets or hierarchical structures (like the set of all used synapse parameters) in case of the *bio graph*, or hardware units like the HICANN network cores in case of the *hardware graph*. Synaptic connections between neurons or the assignment of parameters are represented by so-called *named edges*. Hierarchical dependencies, e.g. the relation of a neuron to be part of a HICANN, are indicated via *hierarchical edges*.
Then, a *mapping algorithm* connects the nodes of both graphs via so-called *hyper edges*, describing the actual mapping process. The rating of 'quality' of different maps is accomplished via a *cost function*, specified by the experimenter.

A detailed presentation of the Graph Model can be found in [*Wendt et al.*, 2008].

[5] Hochparallele VLSI-Systeme und Neuromikroelektronik, Technische Universität Dresden
[6] Electronic Vision(s), Ruprecht-Karls-Universität Heidelberg

III Models of Organization and Computation in Neural Networks

Typical tasks for neural networks are perception and processing of external stimuli applied to the network. Finally, this process should lead to decisions and initiate actions. This chapter addresses two paradigms that have been proposed in order to perform such tasks:

- The concept of *Liquid Computing* allows manifold computational operations on arbitrary input streams using generic network architectures. (section III.4)
- A hierarchic structure of *Synfire Chains* has been proposed to classify complex objects and time intervals or to provide clean switching between selected actions. (section III.1)

As these concepts are inspired by measurements of the cortex, it is reasonable to provide some general network properties found in biology:

In vivo experiments have shown that in large, active networks synaptic conductance often exceeds the neurons' inherent leak conductance. This so-called *High-Conductance-State* results in fundamental consequences regarding the information processing of individual neurons that should be considered when modeling and investigating neural circuits (see section III.2).

In general, cortical architectures are likely to compensate perturbing influences of erroneous components and background noise. Regarding *analog hardware emulators* like the FACETS Stage 1 Hardware, such compensative mechanisms can counteract variations of hardware devices due to the production process, particularly those that cannot be reduced by calibration algorithms (see section III.3).

III.1 Synfire Chains

When a neuron gets excited by many pre-synaptic neurons simultaneously, it is likely to emit a spike. So-called *synfire chains* are feed-forward networks consisting of a chain of groups, each projecting to the following group. These networks can sustain the propagation of synchronous pulse packets [*Abeles*, 1991] which have been found to withstand dispersion under appropriate conditions [*Diesmann et al.*, 1999], [*Gewaltig et al.*, 2001].

Synfire chains have been proposed to represent primitives in the process of object recognition [*Bienenstock*, 1995]: If multiple interconnected primitives are stimulated at the same time, their chains can compose a wider and more stable synfire chain, representing a more complex object.
As shown in [*Haß et al.*, 2008] synfire chains can also be used to identify time intervals. Furthermore, [**?**] demonstrate, how a set of interconnected, closed-loop[1] synfire chains perform random scribbling.

A stable, closed-loop synfire chain has been set up on the FACETS Stage 1 Hardware, as described in section VI.1.

III.2 High Conductance States

As stated above, in awake animals neurons experience considerable synaptic input most of the time [*Destexhe et al.*, 2003]. In simulations this background stimulus can be modeled as additional, statistical conductance of the ion channels [*Destexhe et al.*, 2001]. Effectively, this *High-Conductance-State* results in a shorter membrane time constant compared to the resting state and yields a *probabilistic dependency*, whether a spike is emitted under a given stimulus. Especially, a weak input, that would not cause a resting neuron to spike, reveals a non-zero probability to trigger a post-synaptic spike [*Hô and Destexhe*, 2000]. Furthermore, the reduced time constant allows the membrane potential to follow changes in the input even for high frequencies, resulting in an increased *temporal resolution* of the neuron [*Destexhe et al.*, 2003]. Thus, changing a neuron's state between 'resting' and 'high-conductance' allows perceiving the neuron as *integrator* and *coincidence detector*, respectively [*Rudolph and Destexhe*, 2003].

A proposal on approximating high-conductance-states within the integrate-and-fire neuron model considering the limited number of inputs of the FACETS Stage 1 Hardware is presented in section VI.2.

III.3 Self-stabilizing Networks

Cortical neurons of awake animals exhibits a spontaneous firing activity of typically about 5Hz - 20Hz [*Baddeley et al.*, 1997; *Steriade*, 2001; *Steriade et al.*, 2001]. Although this omnipresent activity is often assumed when modeling and simulating neural circuits, it has been stated that it is a challenging task to set up networks featuring a low but stable activity over a wide range of external stimuli [*Abeles*, 1991]. An auspicious approach to self-stabilizing network architectures was introduced by [*Sussillo et al.*, 2007] and will be presented in the following. It employs a phenomenological model of short-term plasticity proposed by [*Markram et al.*, 1998] and utilizes – in accordance with [*Thomson*, 1997] – that the dynamics of a synapse depend on the type of the pre- and post-synaptic

[1] closed-loop synfire chain: The last layer projects on the first layer, forming a closed loop.

neuron. *"This self-tuning principle enables neural circuits to respond to external perturbations with a characteristic transient response in the firing rate of excitatory neurons, and then returns to its previous firing rate within a few 100 ms back into a given target range."* [*Sussillo et al.*, 2007].

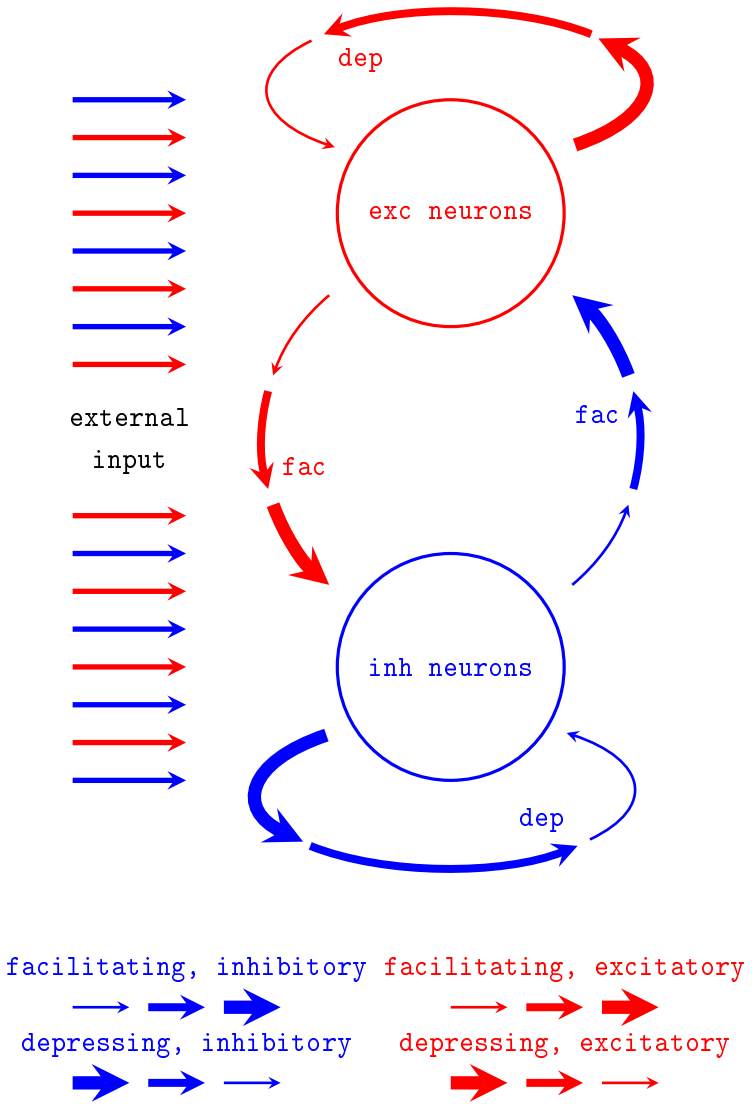


Figure III.1: *Schematic of the self-stabilizing network architecture proposed by [Sussillo et al., 2007].*

A schematic of the investigated network architecture is shown in figure III.1. Two populations of neurons, an excitatory and an inhibitory one, receive uncorrelated, external

stimuli. The rate of the input spike trains can vary over time. Each population projects to both itself and the other population. The parameters of the synapses are chosen such that

- synapses between neurons of the same population are *depressing*, while
- synapses connecting neurons of different populations are *facilitating*

over a wide range of pre-synaptic firing frequency. Thus, on increasing total network activity, the recurrent excitation acting on the excitatory population and the recurrent inhibition acting on the inhibitory population are damped, while inter-population connections are strengthened, favoring inhibitory activity throughout the network. Alike, for decreasing total network activity, the presented connectivity pattern privileges the emission of excitatory spikes. Overall, the network tends to both moderate very high and avoid very low activity and, thus, exhibits a self-stabilizing behavior.
While such networks were proven to almost completely compensate the impact of input variations on the firing rate of the excitatory population *on long time scales*, i.e. some hundred milliseconds or longer, external perturbations still significantly affected network activity for about 50 - 100 msec. Hence, such circuits preserve their ability to integrate external inputs into computation [*Sussillo et al.*, 2007].

Since the FACETS Stage 1 Hardware provides a similar short-term plasticity mechanism (see section I.3), it is feasible to apply the above concepts to the hardware in order to counteract inevitable perturbations of such an analog device.[2] For their simulations [*Sussillo et al.*, 2007] used networks of 5000 neurons. As each *Spikey* chip only holds 384 neurons, it needs to be verified whether small-sized networks are capable of exploiting this paradigm (see section VI.3).

III.4 Liquid Computing

The computational paradigm of the *liquid state machine* (LSM) was proposed and analyzed by [*Maass et al.*, 2002], and will be shortly described with respect to spiking neural networks[3] (see figure III.2) in the following. Independently, [*Jaeger*, 2001] introduced a similar concept, the *echo state* approach.

The state of a randomly, sparsely connected network consisting of N neurons – the so-called *liquid* – can be described as a high-dimensional vector $x(t) \in \mathbb{R}^N$ varying over time. The state $x(t)$ consists of the N states $x_i(t)$ of the liquid neurons.[4] When the liquid gets externally stimulated by one or more spike trains $u(\cdot)$ projecting on different subsets of the liquid, the input stream is mapped to a high-dimensional state vector trajectory $x(\cdot)$. Due to the recurrent connections, $x(t)$ depends on both the previous and the current input, revealing a *fading memory*. Usually, the liquid does not converge

[2] Several examples for inherent, hardware specific perturbations can be found in chapter IV.
[3] The liquid state machine is an abstract concept that does not necessarily require neural networks.
[4] A suitable state $x_i(t)$ can be defined by adding 1 for every spike while decaying with a typical membrane time constant τ: $\dot{x}_i = -x_i/\tau + \delta(t - t_{\text{spike}})$

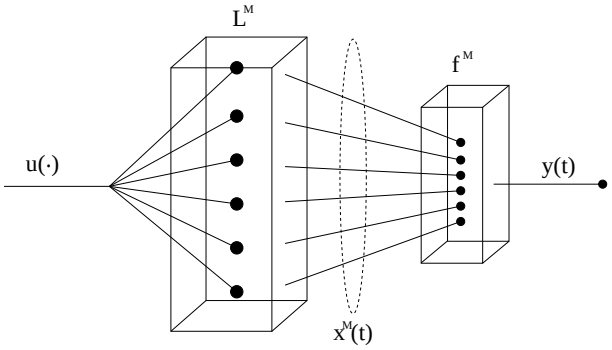


Figure III.2: *Schematic of the liquid state machine, according to [Maass et al., 2002]. A recurrent network (the liquid filter L) and a subsequent readout function f transform the input stream $u(\cdot)$ into an output function $y(t)$.*

to an attractor state, but describes a trajectory of transient states over time. If the connectivity of the recurrent network is chosen appropriately, different input streams are clearly separated in the state space, while moderately jittering the input will result in similar trajectories.
As stated in [*Maass et al.*, 2004] a sufficiently complex liquid performs manifold, non-linear operations on the input when mapping the input stream into the high-dimensional state space. The resulting 'computational primitives', represented by the neuron states $x_i(\cdot)$, are accessible to a *readout* that needs to be trained for the specific task. [*Maass et al.*, 2002] proved that even a memory-less readout function $f : \mathbb{R}^N \to \mathbb{R}$ can theoretically extract any information on the elapsed input stream from the current state vector $x(t) \in \mathbb{R}^N$ that can possibly be provided by a system with fading memory. In this sense, the *liquid state machine*, consisting of the liquid and a readout function, possesses *universal computational power*.
While the readout function f is trained for a specific task with supervised learning rules, the liquid is a generic network that is widely independent of the designated application. This allows multiple readout functions to be served by one liquid in parallel. In practice, it is often sufficient to use small feed-forward perceptron[5] networks as readout functions in order to perform complex tasks like pattern recognition, spike coincidence detection or calculating the time integral of firing rates.
Furthermore, computations are performed in real time, enabling the LSM e.g. to recognize a spike pattern and to provide this result to another computational unit even before the input stream was completely injected.[6]

The question how a recurrent network should be set up in order to provide an optimal map $u(\cdot) \mapsto x(\cdot)$ is subject to ongoing research. E.g. computer simulations comparing

[5]The perceptron describes an early neuron model introduced in [*Rosenblatt*, 1958].
[6]Obviously, this example assumes that the already injected part of the spike pattern allows an unambiguous identification of the pattern.

liquids implementing static synapses with those using dynamic synapses indicate an important computational role of synapse dynamics [*Maass et al.*, 2002] – in accordance with earlier analytical studies [*Maass and Sontag*, 2000]. A fast neural emulator like the FACETS Hardware could be utilized for systematic analyses of the computational capability of different recurrent circuit architectures.

III.4.1 High-Dimensional Attractors

An improvement of the liquid state machine introducing *persistent memory* was proposed by [*Maass et al.*, 2007]. Beside training the memory-less readout function, some neurons inside the recurrent network – so-called *high-dimensional attractors* – are trained to perform a specific task when triggered by the input (e.g. to generate a specific firing pattern) or to act as memory unit (e.g. indicate which input stream bursted most recently). The training was accomplished with a subsequent supervised learning rule named 'teacher forcing'. A such trained memory that involves only a couple of neurons was shown to be capable of storing information over long time spans. Inevitably, this results in a slightly restrained state space, as a small number of 'dimensions' is fixed to a specific value representing the attractor state. The other neurons of the liquid still process the input streams without hindrance, preserving a high-dimensional state space and, thus, the computational power of the LSM.

IV Investigation and Minimization of Analog Imperfections

Regarding the requirement for homogeneity of the neural substrate on the unit level, the uncalibrated FACETS Stage 1 Hardware violates this postulate in manifold ways. Some inaccuracies are unavoidable when producing a physically implemented neural emulator (i.e. as a microelectronic realization) and are allowed for by design. Others are unintended and have been discovered over time.
This chapter discusses an assortment of such inhomogeneities, partly discovered by the author. It is aimed for an understanding of the underlying effects on a microelectronic level, in order to minimize their influence on the network behavior via additional parameter adjustment or temporary workarounds.

Section IV.1 investigates a subtle system-wide interference of the different hardware components. The following three sections address variations of the synaptic efficacy (section IV.2 and section IV.3) and of the hardware neurons (section IV.4) due to the production process. Concepts for determining and reducing these fluctuations are developed. In section IV.5 the short term plasticity mechanism of the hardware synapses is analyzed. As it will be found that the desired configuration is hindered by insufficient parameter ranges, a workaround is presented, which allows an adequate operation. Finally, the dependency of the hardware behavior on the chip temperature is investigated in section IV.6.

IV.1 Synaptic Background Conductance

It was observed that the *neuron resting potential* does not only depend on the preset V_{rest} voltage (see section I.2) but also on the configuration of the entire network. A systematic investigation revealed that a neuron's observed resting potential depends on

- the number of external inputs connected to the neuron,
- the hardware weight of the established connections and
- whether the synapses are excitatory or inhibitory,

but is independent of

- the maximum synapse driver strength `drviout` and
- the number of processed spikes.

This suggests that there exists a constant and permanent *synaptic leakage conductance* caused by a too high V_{out} voltage of the synapse drivers even while there are no APs processed (see section I.3). Indeed, such a permanent conductance is existent by design as the *synapses* transform V_{out} into $w \cdot g(V_{out})$ exponentially yielding an always non-zero conductance between the membrane and the reversal potentials. But the observed influence on the neuron membrane exceeds the expected amount by far: *Synaptic leakage conductance* can easily cause an otherwise resting neuron to fire continuously, as shown in figure IV.1.

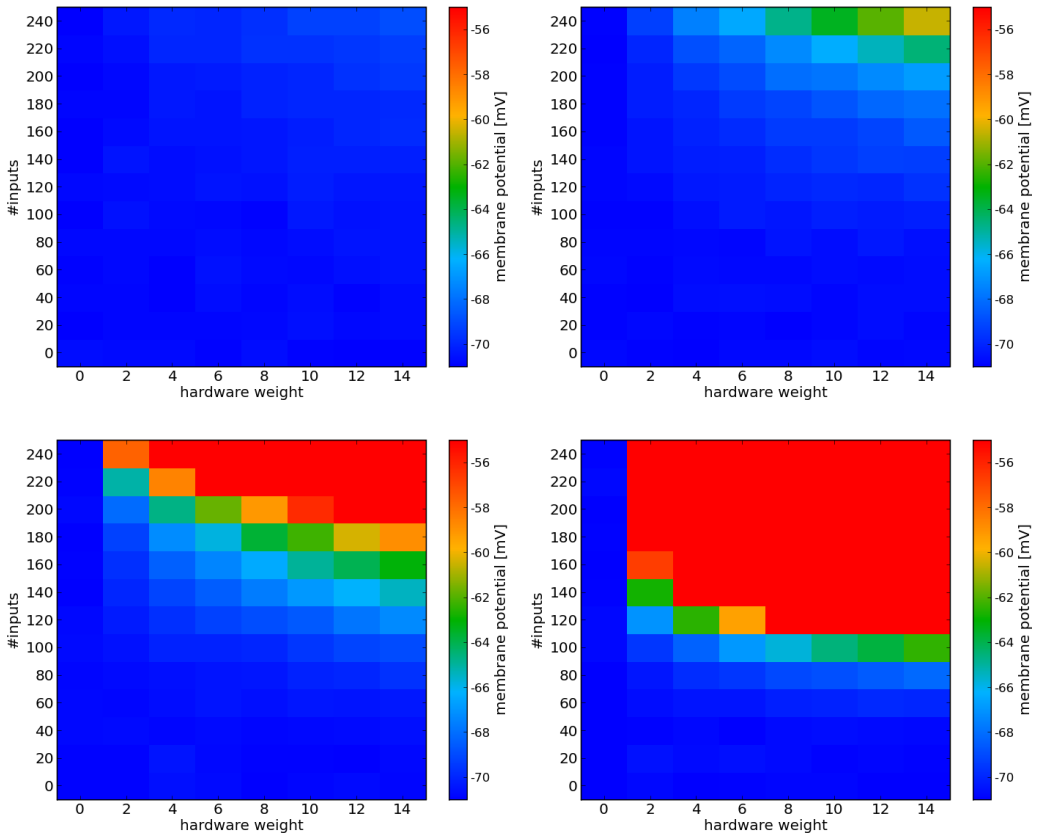


Figure IV.1: *Shifting of the neuron resting potential due to synaptic leakage conductance.* `#inputs` *synapse drivers are configured with* $\texttt{I}_{\texttt{fall}} \neq 0$ *and synaptically connected to a neuron with the discrete excitatory weight* w*. The measurement was taken on an arbitrary neuron of Spikey 18 exhibiting an unmodified resting potential of* $V_{rest} = -70.8\,\mathrm{mV}$*. The plots show the effect with different values of* $\texttt{I}_{\texttt{fall}}$*. From upper left to lower right:* $0.15\,\mu\mathrm{A}$*,* $0.3\,\mu\mathrm{A}$*,* $0.5\,\mu\mathrm{A}$*,* $1.0\,\mu\mathrm{A}$*. For* $\texttt{I}_{\texttt{fall}} = 0\,\mathrm{A}$ *the membrane potential remains completely unchanged. The color-bar is cropped at* $V_{mem} = -55\,\mathrm{mV}$*, a typical neuron threshold voltage. With disabled spike mechanism the highest membrane voltage observed was* $V_{mem} = -36.0\,\mathrm{mV}$*.*

In November 2008 measurement taken by Dr. Andreas Grübel and the author revealed that the synaptic resting potential VREST provided by the *Recha-DAC* located outside the chip (see section I.4) is not stable at a low voltage. Caused by a yet unknown malfunction a current on the VREST line of up to 4.3 mA generates a voltage due to several resistors[1] between the DAC and the synapse driver circuitry. The effect of a raised VREST voltage on the *synaptic conductance* is illustrated in figure IV.2.

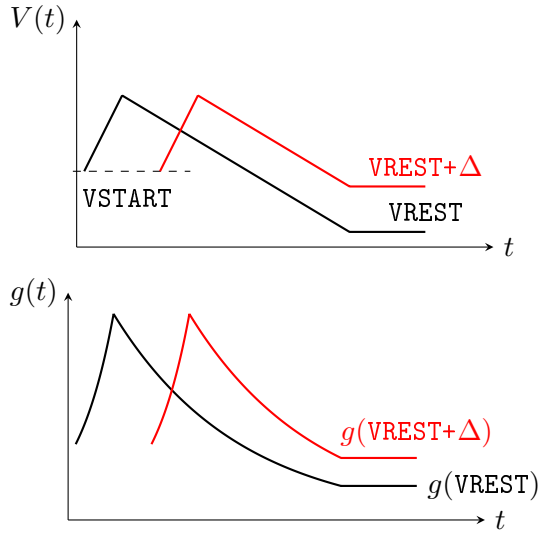


Figure IV.2: *Schematic of the linear voltage ramp generated by a synapse driver and the resulting synaptic conductance course. The red graphs show the effect of an instable VREST voltage that has been raised by Δ due to erroneous leakage currents. Even a 'resting' synapse driver induces a noticeable permanent synaptic conductance. Additionally, the minimum synaptic efficacy of dynamic synapses determined by the difference VSTART - VREST is reduced (see section IV.5).*

Obviously, the effect is determined by $\mathtt{I_{fall}}$, the current controlling the synaptic time constant τ_{fall} (see figure IV.1). For $\mathtt{I_{fall}} = 0\,\mathrm{A}$ the membrane potential is not affected at all. Synapse drivers operating the left or the right synapse array distort VREST equally (not shown). The current observed on the VREST line exceeds the expected current of $\mathtt{\#inputs} \cdot \mathtt{I_{fall}}$ by about one order of magnitude. Other synapse driver parameters like $\mathtt{I_{rise}}$, $\mathtt{I_{gmax}}$, $\mathtt{V_{plb}}$ as well as all STP parameters do not contribute to the above effect. Further measurements revealing the source of the too high leakage current need to be taken.

Until the VREST voltage can be stabilized on a low value it is recommended to restrict any synapse driver $\mathtt{I_{fall}}$ current to a maximum of 0.15 µA as this value guarantees stable neuron resting potentials. A workaround applied to *Spikey 25* lowering VREST in order to allow a decent operation of *short-term plasticity* is presented in section IV.5.

[1] Mostly, the DAC's source resistance and a protective resistor

IV.2 Synapse Driver Calibration

Due to variations in the production process, synapse drivers reveal a manifold behavior regarding the maximal amplitude g_{max} and the time constants τ_{rise} and τ_{fall} of the generated voltage ramps (see section I.3).
Concerning g_{max} the variances arise from comparator biases of the comparator O_1 (see fig. I.4). The time constants are adjusted via current mirrors involving several transistors. Their mismatch causes variations of the time constants. In order to counterbalance these inhomogeneities and to adjust the characteristics of the synapses in general, all three controlling currents – namely, `drviout`, `drvirise` and `drvifall` – can be set individually for each synapse driver (see also section II.2).
Since neither the synapse driver's voltage course nor the conductance course at the ion channel[2] are observable, a calibration routine must match the effect of a synapse on the membrane potential to the desired conductance course. An opportune method is to simulate the favored synapse behavior with a software simulator and adapt the Hardware synapse to the simulation results.

The calibration process is considerably hindered by the capacitance of the vertical wires, transmitting the current from the synapse to the ion channel. Further, in typical networks a neuron experiences numerous conductive connections to the reversal potentials most of the time – altering the neurons effective resting potential V_{rest} and relaxation time constant τ_{mem} (see also section III.2).
Taking these effects into account, previous calibration algorithms stimulated the neuron with several *Poisson spike trains*. Knowing the spike times of each driver, every EPSP can be assigned to its generating driver. In a single trial, individual EPSPs are unrecognizable due the noise of the EPSPs from other synapses. But after stimulating the membrane with numerous different Poisson spike trains, the noise averages out, revealing the separate EPSPs. This method is known as *spike triggered averaging* (STA, for a typical application of this technique see e.g. [*Matsumura et al.*, 1996]. After measuring all drivers' strengths, their `drviout` and τ_{fall} are adjusted. This procedure is repeated, until the desired accuracy is reached.

A simpler method has been proposed by the author, exploiting the *synapse drivers' leakage* described in section IV.1: Almost all synapse drivers get connected to a neuron with high synaptic weights. Without spike input they add a medium and constant conductivity to both reversal potentials. The fraction of excitatory to inhibitory synapses determines the effective resting potential of the neuron. Using this setup, the calibration process can be reduced to single EPSPs from the considered synapse driver. Only the noise of the membrane and the analog readout must be averaged out.
In order to accelerate the calibration routine, multiple synapse drivers[3] process a spike in the same trial, but at different points in time. Furthermore, the calibration process is

[2] More precisely, the hardware equivalent to the biological ion channels, defined in section I.3.
[3] typically 8 or 16

shortened without losing precision, as `drviout` is adjusted just roughly in the beginning, increasing precision with each trial by incrementing the number of averaged repetitions. The presented algorithm has been implemented by the author (filename: `drviout-Calib.py`). The current version only alters `drviout`, while `drvirise` and `drvifall` are set to fixed values, and uses the integral of the EPSP as measure. Further, the calibration uses only one (arbitrary) neuron. Thus, variances of the EPSP caused by the neuron do not average out. The impact of this effect is discussed in section IV.3. Future versions could improve performance by averaging the calibration factors obtained from different neurons.

Comparison of Performance

In order to evaluate the precision, achieved by the different calibration algorithms, Daniel Brüderle implemented a tool to analyze the EPSP-integrals. It is based on his calibration routine using *spike triggered averaging*. One chip's (Spikey III #18) left synapse driver block was calibrated with both calibration algorithms, using the same reference neuron. The evaluation was run on this neuron, too, in order to eliminate *variations caused by different neurons* (see section IV.3).
The calibration exploiting the *synapse drivers' leakage* yielded a relative standard deviation of about 15% over all synapse drivers' EPSPs and took 195 min. The calibration based on *STA* resulted in a relative standard deviation of about 20% and took 199 min. Both algorithms allow a satisfying calibration of the synapse drivers, approaching the limit derived in section IV.3. A total duration of 2×3 hours is acceptable to calibrate both synapse driver blocks. The algorithm proposed by the author performed slightly better, but might lack on compatibility with future Hardware revisions, since it uses an unintended 'feature'.

Remark: The comparison was carried out before the source of error of the *synaptic leakage conductance* was discovered in November 2008 (see section IV.1). Hence, both calibration routines had to deal with variable membrane resting potentials. Presumably, a repetition would yield an increased performance of both calibration methods. The approach of the author was still applicable despite of a reduced synaptic leakage conductance with $\mathtt{I_{fall}}$ limited to $0.15\,\mu$A.

IV.3 Exploring the Limits for Calibration Algorithms

When counterbalancing variances of the synapse drivers, the question arises, what level of uniformity – regarding the integrals of the EPSPs – can be achieved.
Basically, four components contribute to the actual membrane potential seen on the oscilloscope, whenever an EPSP occurs[4]. Each of them adds specific variations and statistical errors.

[4] Assuming an idealized setup with only one EPSP on a resting neuron.

(1) **The synapse driver** creates the linear voltage ramp transmitted to the synapses (see sections I.3 and IV.2). (*driver specific*)

(2) **The synapse** transforms the voltage into an exponential current (see section I.3). (*individual for each synapse*)

(3) **The vertical conductor, the ion channel[5] and the neuron's capacitance** determine the impact of the current on the membrane potential. (*neuron specific*)

(4) **The analog readout** connects the membrane to the oscilloscope. (*neuron specific*)

(5) **All components** are subject to *statistical noise*.

Minimization of (1) has been discussed in the previous section IV.2.
Systematic readout errors (4) could – in principle – be measured by setting a neuron to a fixed, predefined voltage (for instance a resting potential) and observing its voltage on the oscilloscope. Correction factors obtained in this way could adjust latter measures. The systematic readout errors caused by the eight analog readout channels, are being measured and included in the software flow by Daniel Brüderle, while this thesis is being written. Their contribution to the readout error was found to outrange neuron-wise effects by about factor 10.
Statistical errors due to noise (5) can be reduced arbitrarily via numerous repetitions of the same setup.
Genuine neuron specific variations (3) can only be counteracted statistically in larger networks by multiplying all incoming synapse weights with a neuron specific *weight factor*. This approach has been suggested and implemented by Daniel Brüderle (see also section II.2).
Only the variations of the synapses w themselves (2) cannot be reduced, since their discrete nature prevents stepless corrections. Hence, this variation sets the benchmark for all calibration algorithms concerning items (1) and (3)[6].

The author proposed and implemented the following method to estimate the influence of the *synaptic variance* (2) on the EPSP-integral. Based upon an adequate calibration of the synapse drivers using `drvioutCalib.py`[7] presented in section IV.2 and a calibration of the membrane leakage conductance g_{leak} using `doCalibration_tauMem.py` by Daniel Brüderle, all $256 \times 192 = 49152$ synapses of a synapse array are surveyed: Each neuron is set to a predefined effective resting potential via *leakage conductance* as applied in the synapse driver calibration. Then, all 256 synapse drivers excite each neuron with one EPSP of the same discrete weight w.[8] The integrals of the EPSPs are measured.

[5] The hardware equivalent to the biological ion channels, presented in section I.3.
[6] Reducing the readout errors (4) does not effect the network's behavior.
[7] The actual precision of the calibration is of no significance, since the synapses of each driver are analyzed separately.
[8] In detail, the neurons are stimulated and analyzed one by one. 32 EPSPs from different drivers are excited per trial at different points in time. The membrane potential is averaged over 10 identical trials in order to reduce statistical noise (5). This process is repeated until all pairs (driver, neuron) have been measured.

In order to remove the *systematic neuron specific errors* (3) and (4), all EPSP-integrals of *the same neuron* are divided by their common mean, multiplicatively normalizing them to 1. The standard deviation of these normalized EPSP-integrals of one single *synapse driver* (again normalized to their common mean) is a measure of the relative variation caused by the synapses (2). By averaging the relative standard deviations of all drivers the certainty of the estimate is improved.
Finally, this estimate still contains the (already reduced) error due to statistical noise (see footnote). This error is determined separately (`stdOfOnePsp.py`) based on eight arbitrary synapse drivers and one arbitrary neuron: Ten-times averaged EPSP-integrals are gathered 50 times for each driver. The mean of the standard deviations of the drivers' EPSP-integrals is an estimate for the noise level of the readout.

The measurements described above have been carried out for the left synapse array of Spikey 25. The target value of the synapse driver calibration was $\int (V(t) - V_{rest})\, dt = 1\,\mathrm{V} \cdot \mathrm{ms}$, given in biological time and hardware voltage. The unnormalized EPSP-integrals are shown in figure IV.3, the neuron-wise normalized EPSPs in figure IV.4.

The normalized EPSP-integrals (excited by the same synapse driver) have a mean relative standard deviation of 13.46%[9].
The noise of the underlying measurement has been determined to be 8.0%.
Thus, the synapse inherent lower bound for the relative standard deviation after all calibrations concerning (1) and (3) is estimated to be

$$\sqrt{(13.46\%)^2 - (8.0\%)^2} = 10.8\%$$

IV.4 Neuron Threshold Variation

For single-neuron experiments the resting potential V_{rest} and the threshold voltage V_{thresh} can be adjusted at will. But since both voltages are shared by 96 neurons (see section I.5), discrepancies of single neurons (resulting from fabrication process variations) can hardly be counteracted in larger networks. A cause for inhomogeneities of V_{rest} has been addressed in section IV.1. The author also discovered serious variations of the threshold voltage V_{thesh}, which are discussed in this section.

The second version of the FACETS Stage I Hardware (Spikey II) suffered from a dramatic instability in many parameters (see section I.4 and [*Müller*, 2008]), affecting (among others) the threshold voltages and concealing other reasons of neuron inhomogeneity. Spikey III solves the problem of drifting voltages and currents, revealing other sources of error. This explains the late formulation of the following conclusion:

[9] Malfunctioning or obviously faultily measured neurons have been sorted out. About 10% in total.

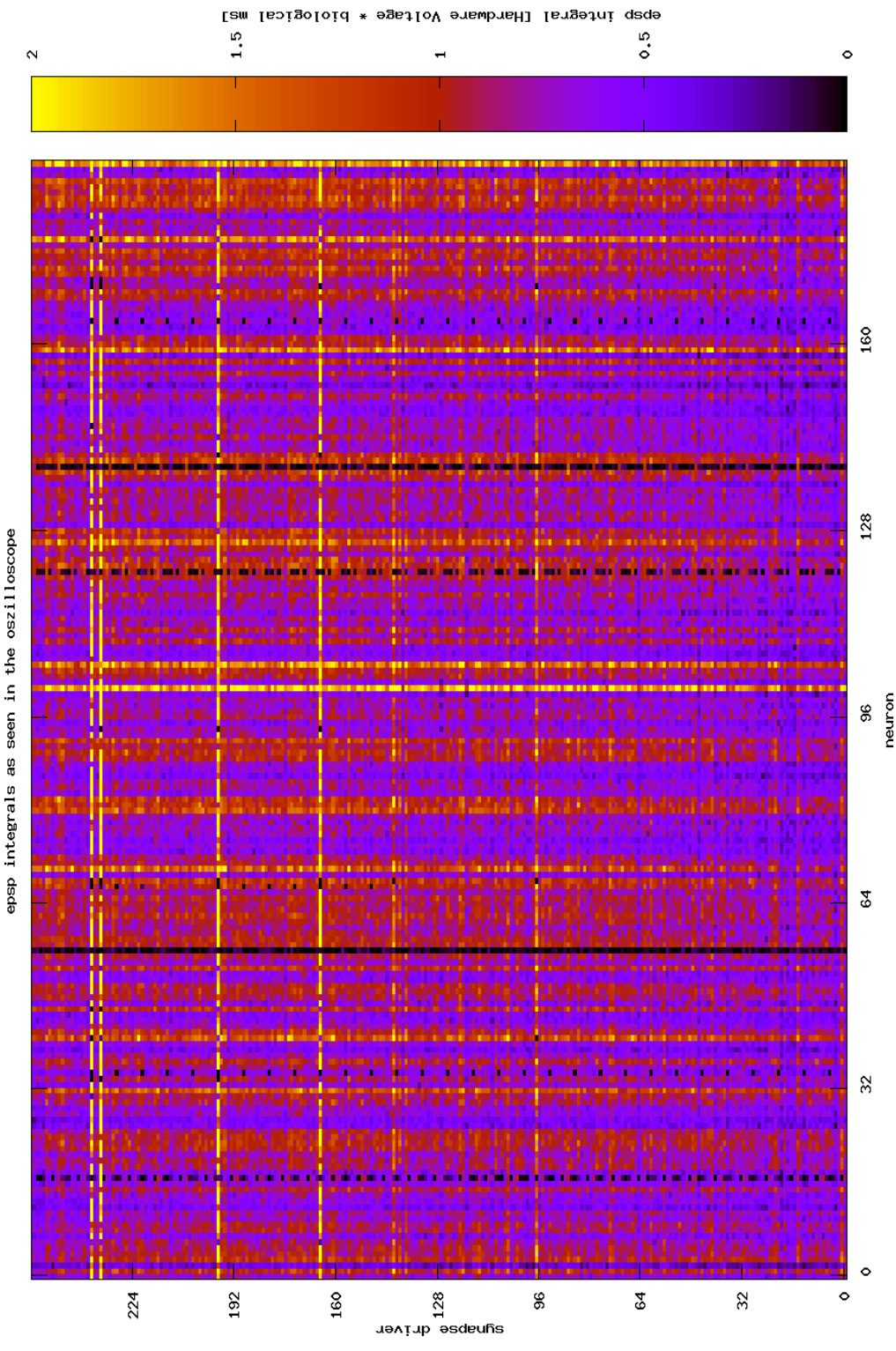


Figure IV.3: *EPSP-integrals as measured on Spikey 25. The synapse driver calibration is based on neuron 58 (for better orientation: the 'black' column is neuron 56). Neuron specific effects are evident. Furthermore, some synapse drivers are generally too weak or too strong, respectively. Some neurons show almost no reaction to the stimulus. The vertical, regular black patterns originate from errors during the measurement: A strong EPSP on a neuron with low threshold (see section IV.4) can cause the neuron to spike. Since multiple drivers are measured simultaneously, such a spike also disarranges the membrane potential of the following EPSPs.*

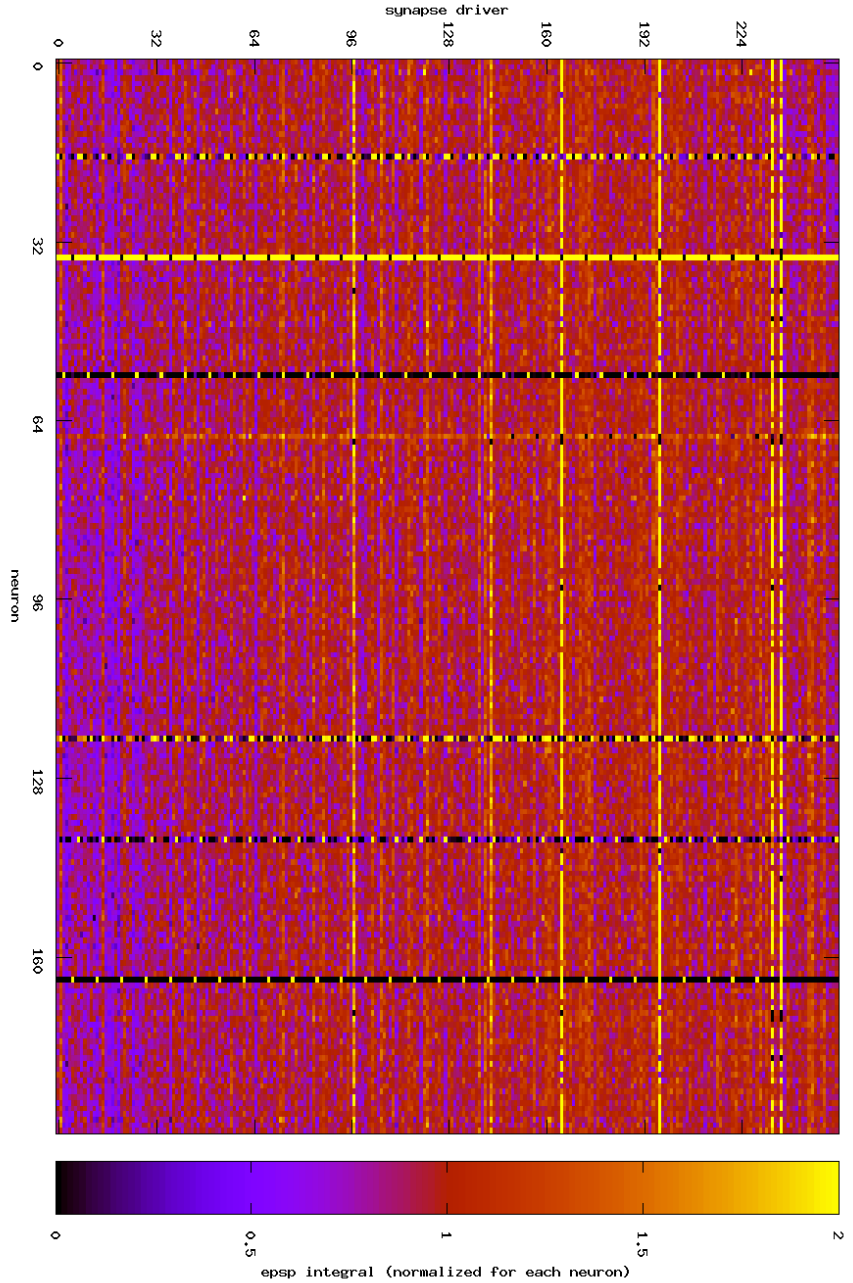


Figure IV.4: *Neuron-wise normalized EPSP-integrals as measured on Spikey 25. For each synapse driver, the variance of the EPSPs is caused by synapse specific variations and statistical noise.*

Many neurons lack either an adjustable threshold or a reliable reset mechanism.

As remarked in section I.2, the bias current `icb` plays a key role for spike emission and membrane reset. `icb` can be set individually for each neuron in a range 0 to 2.5 μA.[10] The threshold comparator contains, among others, two transistors, in order to compare the membrane potential V_{mem} to the preset threshold V_{thresh}. Due to variations in the production process, the transistors do not perfectly match. This results in a comparator bias ΔV_{thresh}, effectively shifting V_{thresh} and, hence, in a variety of neuron specific threshold voltages, that can be either higher or lower than the preset value. `icb` provides the bias current for this comparator circuits: The higher `icb` is set, the lower ΔV_{thresh} becomes [*Schemmel*, 2008].
On the other hand, `icb` also controls the refractory period τ_{refrac}. A high `icb` corresponds to short a τ_{refrac}. In hardware the reset mechanism and the refractory period are combined: When emitting a spike, the neuron gets connected to the reset potential V_{reset} for the period τ_{refrac} with high conductivity. Hence, the refractory period needs to be long enough to discharge the neuron's capacitor (at least close) to V_{reset}. If `icb` is set too high, V_{mem} might even stay above V_{thresh}, locking the neuron in a state that prohibits any further dynamics.

Before these effects were understood, `icb` was a global parameter with a fixed value for all neurons. It was set to 0.2 μA – a rather low value. In large networks this caused some neurons to spike with very high rates without any external stimuli, since their resting potential V_{rest} was above the effective threshold $V_{thresh} + \Delta V_{thresh}$. In recurrent networks such neurons distort the expected network behavior to an extent that makes quantitative studies impossible.
In order to quantify the variance of effective threshold voltages, the threshold was set to $V_{thresh} = -57\,\text{mV}$ for the left neuron block (192 neurons) of Spikey 25. Then, the resting potential V_{rest} was swept over a wide range. A schematic is shown in figure IV.5. The measured neurons' firing rates are presented in the left plot of figure IV.6. The standard deviation of the effective threshold voltages is $\sigma(V_{thresh}) = 4.5\,\text{mV}$. 43 neurons reveal a shift of more than 5 mV.[11]
A more extensive investigation of the effects described above is shown in figure IV.7, exemplary for neuron 256 of Spikey 25. This neuron exhibits a typical behavior for a neuron with negative ΔV_{thresh}: For a fixed difference $V_{thresh} - V_{rest} = 1\,\text{mV}$ and low `icb` the neuron spikes constantly. As long as the reset period τ_{refrac} is long enough to pull down the membrane close to V_{reset}, the spike rate remains on an almost constant level. While increasing `icb` (thus, shortening τ_{refrac}), at some point the reset mechanism does not manage to pull down the membrane potential completely. Thus, the relaxation towards V_{rest} with time constant $\tau_{mem} = C_m/g_{leak}$ exceeds the threshold earlier, resulting

[10] More precisely, `icb` is a current generated by the DAC with upper bound `irefdac`/10.
[11] All voltages are given in biological units. In October 2008 the mapping of the biological voltages onto the `vout` voltages changed due to modifications in the `voutbias` currents allowing a larger dynamical range of the hardware voltages. This leads to a reduction of the threshold voltage variation in biological terms.

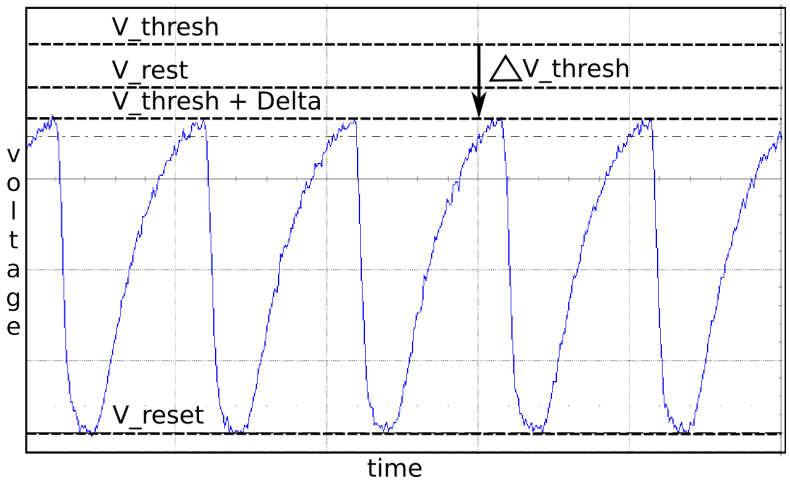


Figure IV.5: *Schematic of threshold voltage deviation and its effect on spiking behavior. Due to the comparator mismatch, the selected threshold and the effective threshold differ. Sweeping V_{rest}, the shift can be detected: For $V_{rest} > V_{thresh} + \Delta V_{thresh}$ the neuron begins to spike repeatedly without any external stimulation.*

in a higher firing rate. Eventually, the reset does not even pull V_{mem} below threshold, causing a locking state and the spike rate to drop to zero. This locking state occurs at lower `icb` as the threshold (and V_{rest}) rises – a correlation that has not yet been understood.

These results suggest to adjust `icb` for each neuron, such that ΔV_{thresh} is reduced, while the reset mechanism still pulls down V_{mem} close to V_{reset}. However, for some neurons, ΔV_{thesh} cannot be reduced to an acceptable extent, before their reset mechanism fails. In order to preserve overall network behavior, such neurons should rather be switched off than be allowed to spike erroneously.

Based upon these facts, the author proposed and implemented a calibration routine (`calibrator_icb_min.py`). Highest priority is that `icb` is high enough to prevent erroneous, constant spiking – either by reducing ΔV_{thresh} to an acceptable extent or by 'switching off' the neuron as it enters a locking state. This condition determines $\texttt{icb}_{\min}$. The central plot of figure IV.6 shows the results of this first calibration routine: It performs well regarding erroneous, unstimulated spiking. The introduction of an upper limit for $\texttt{icb}_{\min}$ (right plot of figure IV.6) is not recommended.
While this thesis is being written, the above algorithm is included in an improved calibration routine by Daniel Brüderle that determines a reduced threshold discrepancy ($\rightarrow\texttt{icb}_{\min}$) and a minimal refractory period ($\rightarrow\texttt{icb}_{\max}$), separately. A future revision of the FACETS Stage 1 Hardware – scheduled for summer 2009 – will separate the comparator bias from the refractory period, allowing to reduce ΔV_{thresh} without affecting

τ_{refrac}. Additionally, a more sophisticated mapping process (section II.3), supporting the exclusion of faulty neurons, is in progress.

IV.5 Short Term Plasticity

When investigating the influence of dynamic synapses on a neural network's behavior, it is essential to provide a comparison to the performance of static synapses. In particular, the respective synaptic weights need to be converted into each other in a meaningful manner.
Furthermore, it is desirable to be in control of the applied STP-parameters. As presented in section I.3, all parameters describing the short-term plasticity mechanism of the FACETS Stage 1 Hardware are adjusted by voltages and currents. Since only the influence of these parameters on the membrane potential can be observed, it is a challenging task to identify e.g. the recovery time constant of the inactive partition.

This section addresses the above issues: After a theoretical analysis of the synapse dynamics, a concept is presented on how Spikey's STP-parameters can be measured. It will be found that the parameter range of the `vout`-voltages does not allow an adequate operation of short-term plasticity. Finally, a workaround alleviating this problem is presented.

IV.5.1 Analysis of Synapse Dynamics

Following [*Sussillo et al.*, 2007], the conversion between *dynamic synapse weights* W_D and *static synapse weights* W_S can be achieved by assuming a regular firing of the presynaptic neuron with rate $\nu = 1/T$: The *(in-)active partition* I (see section I.3) will converge towards a constant value $I_0(\nu)$ resulting in a *corresponding synaptic weight*. This weight allows for a conversion between W_D and W_S.
In case of **depression** it is:

$$W_S = W_D \cdot [1 - \lambda \cdot I_0]$$

In case of **facilitation** it is:

$$W_S = W_D \cdot [1 + \lambda \cdot (I_0 - N)]$$

For $I = I_0(\nu)$, the decay of I (with time constant t_{rec}) outweighs the increment $C \cdot (1 - I)$ after every spike. Hence, the equilibrium condition reads:

$$\begin{aligned} C \cdot (1 - I_0) &\stackrel{!}{=} [I_0 + C \cdot (1 - I_0)] \cdot [1 - \exp(-T/t_{rec})] \\ \Leftrightarrow I_0 &= \frac{C}{\exp(T/t_{rec}) - 1 + C} \end{aligned}$$

As expected, $\lim_{T \to 0} I_0 = 1$ for high firing rates and $\lim_{T \to \infty} I_0 = 0$ for low rates. Figure IV.8 shows the *(in-)active partition at equilibrium* I_0 as well as the corresponding synaptic weights W_S for exemplary parameters. In contrast to the *UDF-model* proposed

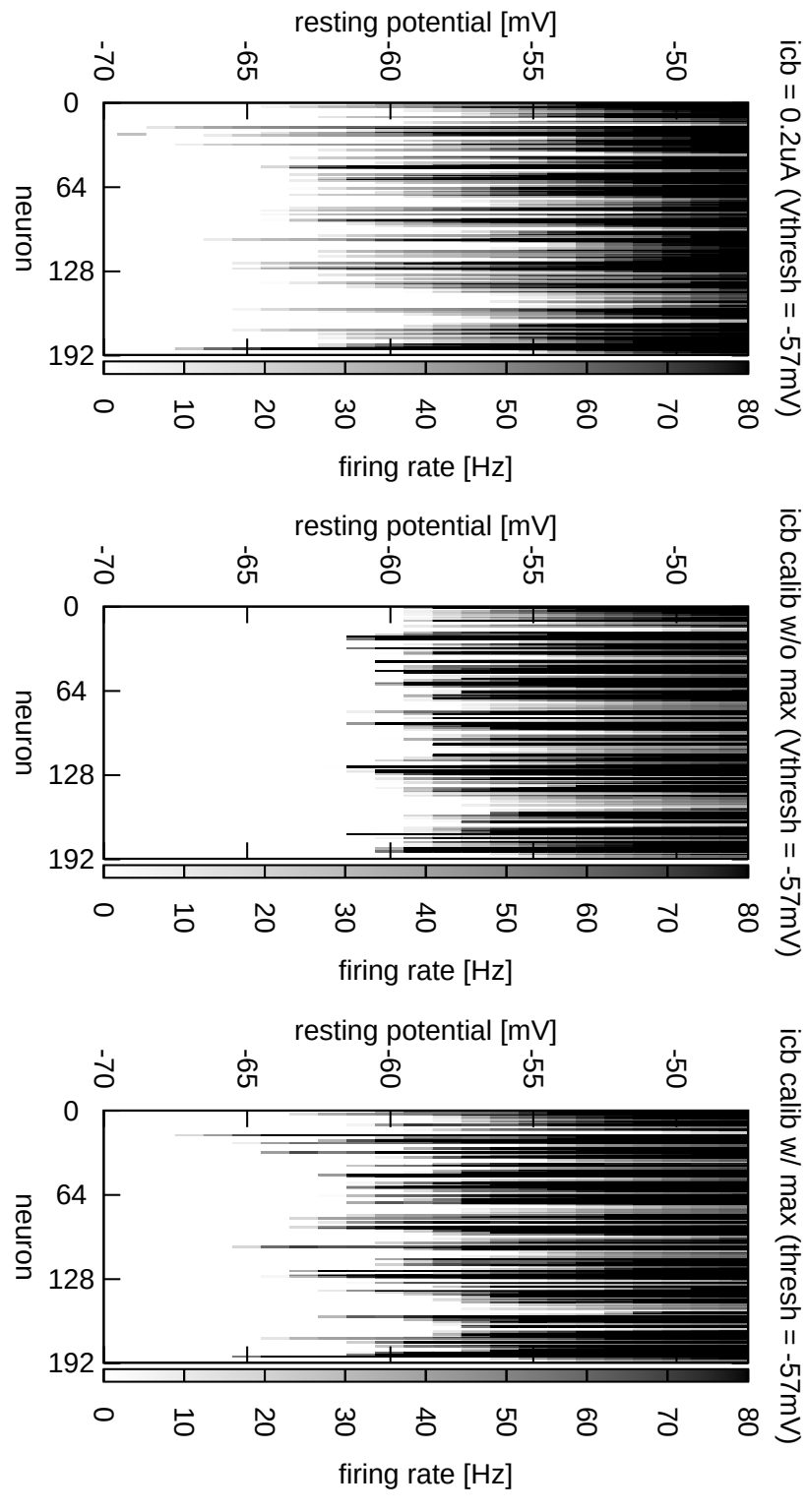


Figure IV.6: *Spike rate of different neurons depending on the resting potential (as measured on Spikey 25), revealing the effective threshold voltages.* left: *with a global comparator bias* `icb` $= 0.2\,\mu$A*;* center: *with individual* `icb` *allowing arbitrarily high biases;* right: *with individual, but limited* `icb`*.*

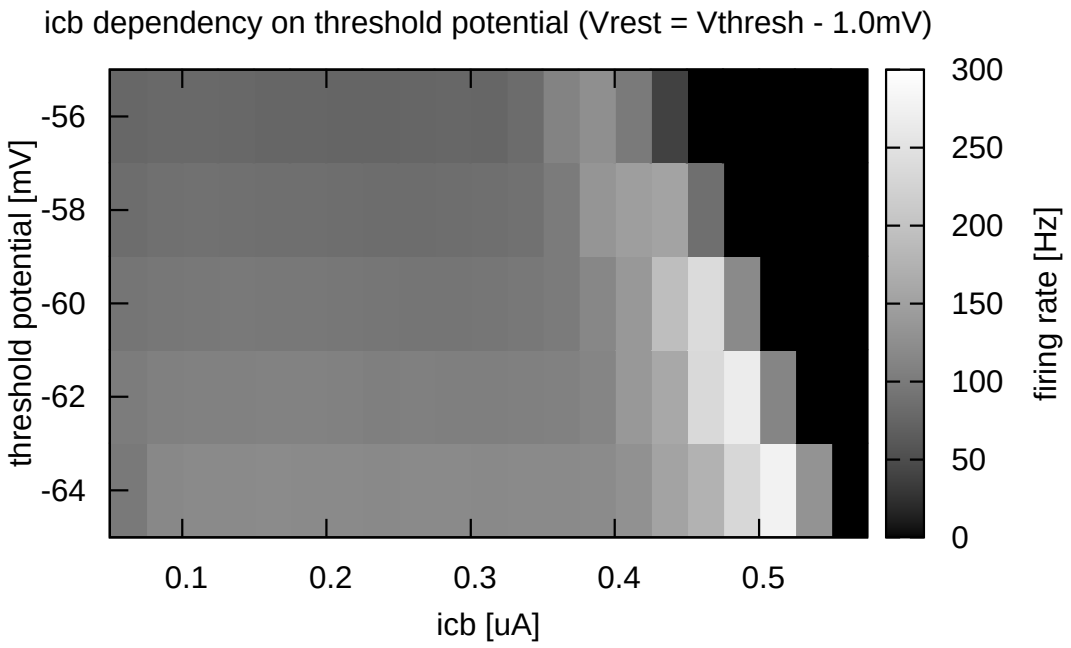


Figure IV.7: *Erroneous spike rate of a typical neuron with lowered threshold (neuron 256 of Spikey 25). For low* `icb` *the reset mechanism works properly. Before the neuron enters a locking state, the spike rate rises, since the membrane potential is set close below threshold after each spike. One notices that the edge of the locking state depends on the preset threshold voltage.*

by [*Markram et al.*, 1998], which the hardware STP-mechanism is based on, the dynamics of the FACETS Stage 1 Hardware do not exhibit a maximum at a certain frequency but are strictly monotonic in- or decreasing, depending on their `mode`.

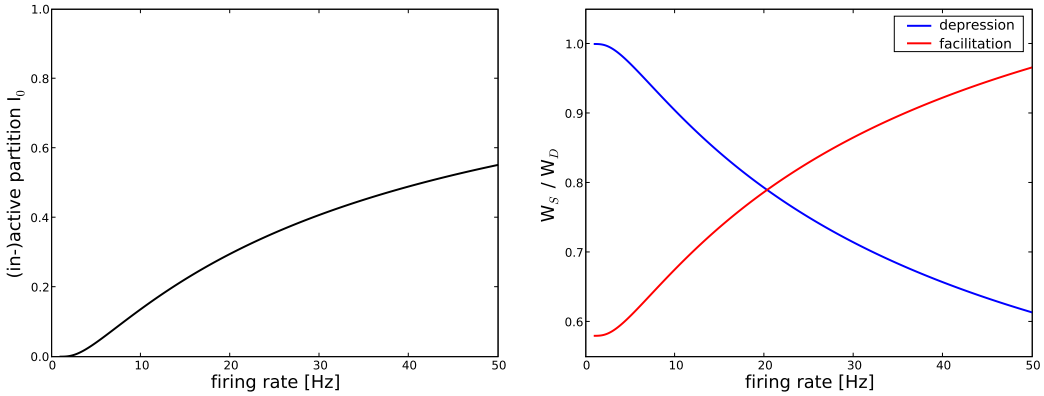


Figure IV.8: left: *The (in-)active partition at equilibrium I_0 with respect to the rate of a regular input.* right: *The corresponding static weight for dynamic synapses in depression and facilitation mode.* parameters used: $C_2 = \frac{3}{8} \cdot C_1$, $t_{rec} = 100\,\text{msec}$, $\lambda = 0.7$ *and* $N = 0.6$.

The largest change of I_0 with respect to the firing rate ν satisfies

$$f(\nu) := \frac{\partial^2 I_0}{\partial \nu^2} = 0.$$

This equation does not have an analytical solution. Numerical calculations of Mihai Petrovici yielded that the solution $\widetilde{\nu}$ of $f(\nu) = 0$ is linear in the reciprocal recovery time constant $1/t_{rec}$ for any fixed step size C. Hence, the product $t_{rec} * \widetilde{\nu}(t_{rec}, C)$ is only a function of C, which is shown in figure IV.9.

IV.5.2 Measurement of Hardware Parameters

As stated above, determining Spikey's STP-parameters is a difficult task, since only the membrane potential V_m is directly accessible by measurement. Taking the expected errors into account, the following paragraphs only aim for a determination of the magnitude of the parameters. Since the considered voltages and currents are *shared* by each 128 *synapse drivers* (see section I.5) a high-precision calibration on a network level is unfeasible, anyway. While the measurement of the gain μ of the STP operation amplifier O_2 was carried out by the author, techniques for gauging t_{rec} have not yet been applied.

The OTA O_2 Gain μ

As presented in section I.3, the STP mechanism does not alter the discrete weight-factor w but the synapse drivers reference current I_{gmax}. Hence, a measurement should

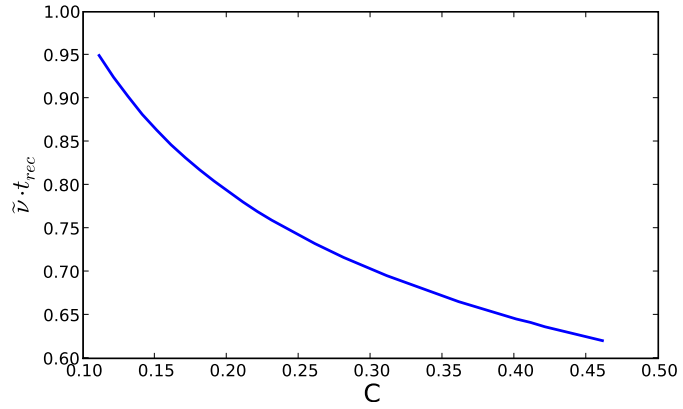


Figure IV.9: *Dependency of the maximum of the derivative of $I_0(\nu)$ with respect to C and t_{rec}. With friendly permission from Mihai Petrovici.*

rather rely on variations of I_{gmax} than on other data. Since the `vout` voltages like $\texttt{V}_{\texttt{max}}$ and $\texttt{V}_{\texttt{fac}}$ exhibit a minimum value of about 0.6 V on Spikey III (see section I.5), it is necessary to switch to facilitation-`mode` in order to apply small voltages to the inputs of the operation amplifier O_2. Large t_{rec} and a large step size $C_2/C_1 = 7/8$ ensure that $I_0 \approx 1$ after few spikes even for moderate input frequencies. Thus, the effective conductance is determined by

$$I_{total} = I_{gmax} \cdot [1 + \mu \cdot (V_{max} - V_{fac})]$$

By recording the resulting membrane potential and adjusting I_{gmax} with *static synapses*, such that the input excites the neuron in a similar manner, the *gain factor* μ can be measured. Based on 4 synapse drivers of Spikey 25, it was found $\mu = (3.0 \pm 0.5)V^{-1}$.

For *depressing synapses* this result means that even the lowest possible value $V_{max} \approx 0.6\,\text{V}$ will push the synaptic efficacy to 0 after few spikes. Correspondingly, for $V_{max} = V_{fac} \approx 0.6\,\text{V}$ the first couple of action potentials transmitted by *facilitating synapses* will leave no trace on the neuron membrane unless the *(in-)active partition* I rises quickly enough, which requires $C_2/C_1 = 7/8$ with a large t_{rec}. This, however, leads to a quick saturation of I close to its maximum value. As a result, both modes lead to an almost 'binary' synaptic behavior, yielding an effective weight of *zero* or *max* most of the time.

Differential Measurement of t_{rec}

The circumstances described above make it difficult to measure the *recovery time constant of the (in-)active partition* t_{rec}. But as soon as the full range of required `vout`-voltages can be attained, the following method should be able to determine the relation between t_{rec} and $\texttt{I}_{\texttt{rec}}$ ($= \texttt{V}_{\texttt{dtc}}$).

Since the fraction C_2/C_1 can be assumed to be rather precise,[12] one can exploit that $\left|\frac{\partial W_S}{\partial \nu}\right| \propto \left|\frac{\partial I_0}{\partial \nu}\right|$ is maximal for $\nu = \widetilde{\nu}$ as shown above. Hence, by varying the input frequency ν and comparing the impact on the membrane to the effect caused by static synapses,[13] $W_S(\nu)$ can be measured and thus also t_{rec}.

IV.5.3 Random Spike Losses

Particularly, during the activation of short term plasticity random spike losses were observed, indicating that, under certain circumstances, some spikes entering the synapse driver are not processed at all. It seems that this error occurs if the duration of the action potential – more precisely the `pre`-signal – is not set up properly. The involved circuity is described in section I.3 and section I.4. The duration of the AP is controlled via $\mathtt{V_{plb}}$ in the software layers. If EPSPs drop out randomly, $\mathtt{V_{plb}}$ might be set either too low or too high.
High values of $\mathtt{V_{plb}}$ cause *short* `pre`-pulses which might not suffice to trigger the synapse driver to start the voltage ramp. Low values of $\mathtt{V_{plb}}$ yield *long* `pre`-pulses that can cause a *locking state* in the circuits generating the `pre`-signal. This state results in a permanent current between `VSTART` and $\mathtt{V_{out}}$ leading to a constant synaptic conductance.
The current setting $\mathtt{V_{plb}} = 0.15\,\mu\mathrm{A}$ supports both static and dynamic synapses.

IV.5.4 A Workaround regarding Internal Voltages

As shown above, restrictions on the configuration range of *Spikey III*'s `vout` voltages lead to a much too strong STP mechanism: A narrow range of possible values for the *(in-)active partition I* covers the whole spectrum of obtainable effective synaptic weights. Two workarounds presented in the following allow for a decent operation, anyhow.

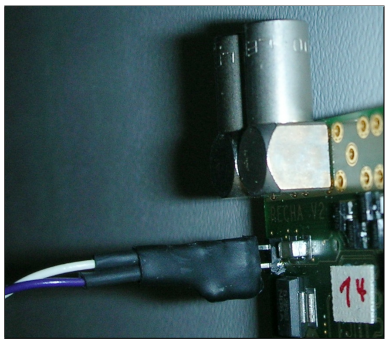


Figure IV.10: *The* `ibtest` *pin is located in the upper left corner of the* Recha *board below the analog readout pins.*

In order to achieve `vout` values below 0.6 V, the `ibtest` pin described in section I.4 can

[12] In addition, averaging the measurement over multiple synapse drivers reduces statistical errors due to variations of the capacitances.
[13] By readjusting $\mathtt{I_{gmax}}$.

be utilized. Usually, it provides *read access* to a *single*, while arbitrary, `vout` voltage. However, the existing wiring can also be exploited to gain *write access* to *multiple* `vout`s in parallel under the restriction that *(a)* all connected voltages will drop to the same value and *(b)* the resulting value cannot be adjusted at will due to the additional resistance of the `ibtest` circuitry.
In practice, a cable is plugged to the `ibtest` pin as shown in figure IV.10. It can be either simply hot-wired or connected to an external *power supply unit* (optionally with reversed voltage). Before multiplexing the `vout`s to the pin the respective `voutbias` currents must be set to zero in order to limit the load on the `ibtest` circuitry. Furthermore, the internal `vout` generation via the DAC should be reduced by setting the requested voltage to zero.
All these issues are covered by the additional *keyword argument* `voltageOnTestPin=[...]` of the `pyNN.run()` function of the `pyNN.hardware.stage1` module. By supplying a list of *indices* of `vout`s to `pyNN.run()`, the respective voltages of both halves of the chip will be connected to `ibtest` during the subsequent configuration process.
For instance, `pyNN.run(1000, voltageOnTestPin=[12, 13])` will provide write access to all four $\mathtt{V_{fac}}$ voltages and then run the network for 1 biological second while the indices [14, 15] would select $\mathtt{V_{stdf}}$.
This workaround must be handled with care since the currents of all connected `vout`s gather in the wire to the pin. The total current, which can be monitored via an amperemeter, must not exceed 3 mA.

The *second workaround* effectively implements a minimum synaptic control current $\mathtt{I_{gmax}}$. As described in section IV.1, *Spikey III*'s global synapse driver resting voltage `VREST` is significantly above 0 V if many synapse drivers are configured even with moderate $\mathtt{I_{fall}}$. In order to reduce this problem the connection between *Recha-DAC* and `VREST` was bypassed on *Spikey 25*, leaving only the internal 10 Ω resistor to induce a voltage that raises `VREST`. This allows operating all 512 $\mathtt{I_{fall}}$ currents at about 0.15 μA with no noticeable change in `VREST`.
This low value of `VREST` guarantees that the initial value of each synaptic course controlled via `VSTART` is above the leakage conductance determined by `VREST` as illustrated in figure IV.2. Each synaptic voltage ramp $V_{out}(t)$ starts at `VSTART` independently of the maximal voltage defined by $\mathtt{I_{gmax}}$. (see section I.3). Even in case of extremely low $\mathtt{I_{gmax}}$ – e.g. for $I \approx 1$ in *depression* `mode` or $I \approx 0$ in *facilitation* `mode` – the synaptic conductance jumps a to noticeable value.
While slightly modifying the intended synapse dynamics, this setup limits the effect of *depressing synapses* at high frequencies. On the other hand, *facilitating synapses* exhibit a non-vanishing synaptic efficacy after longer periods without activity.

Combining the above methods, the following settings have been found to provide a suitable short-term plasticity mechanism on the FACETS Stage 1 Hardware:

- short all $\mathtt{V_{fac}}$ (without an additional power supply unit) via `ibtest`,
- set $\mathtt{V_{stdf}}$ to the lowest possible value via the on-chip DAC,

- restrict $\texttt{I}_{\texttt{fall}}$ to $0.15\,\mu$A,
- set $\texttt{VSTART} = 0.25\,$V and $\texttt{V}_{\texttt{dtc}} = 0.7\,\mu$A,
- use $C_2/C_1 = 1/8$ in depression `mode` and $C_2/C_1 = 3/8$ in facilitation `mode`.

Presumably, this technique will become obsolete in Summer 2009 as *Spikey IV* is expected to provide `vout` voltages close to zero.

IV.6 Temperature Dependency

While using the second revision of the FACETS Stage 1 Hardware, *Spikey II*, it was observed that the same configuration and spike input on the same chip yielded differing results in different trials. Though some level of inaccuracy is taken for granted on an analog emulator, the variation of data acquired on different days exceeded the variation of measurements taken consecutively. This suggested the existence of some source influencing the hardware behavior on time scales of hours or days, potential candidates being changes in the power supply, electro-magnetic fields or fluctuations of chip temperature – caused by either the ambiance, the configuration or network activity.
This section addresses the temperature dependency. After the presentation of the experimental setup possible heat sources are investigated. It will be found that the ambient air has a major influence on chip activity. Therefore, a stabilization of the chip temperature is desirable. Most measurements taken on *Spikey II* should still be valid for its successor *Spikey III*. A comparison between cooled and uncooled hardware devices regarding their thermal stability is presented at the end of this section.

IV.6.1 Experimental Setup

In order to quantify the influence of different sources on the hardware behavior three measures were utilized:

- The *ambient air temperature* was measured with an ordinary digital thermometer[14].
- The *chip temperature* was approximated using a *PT 100* temperature probe being attached to the backside of the *Recha* board. Thermal conductance was achieved by heat-conductive paste and unused metal connectors between *Spikey* and the board. The conversion from the PT 100 resistor to degree Centigrade was carried out by a *Philips PM 2525 multimeter*.
- As measure of *chip activity* the mean firing rate of several non-interconnected neurons being excited with arbitrary but fixed Poisson spike trains was used.

Measurement errors – and, in particular, systematic errors – of the stated *chip temperature* are difficult to determine, since the quality of thermal conductance to the chip is

[14] manufacturer: TFA, Kat. Nr.: 30.1026

unknown. An aggravating factor is that the temperature probe is also inevitably influenced by the air flow of cooling fans. A possible, while elaborate, method to measure the chip temperature with high precision would be the application of a IR camera targeted directly at the chip.
Having said this, all values regarding chip temperature must be accepted with caution and shall only be taken as indicators for changes in temperature.

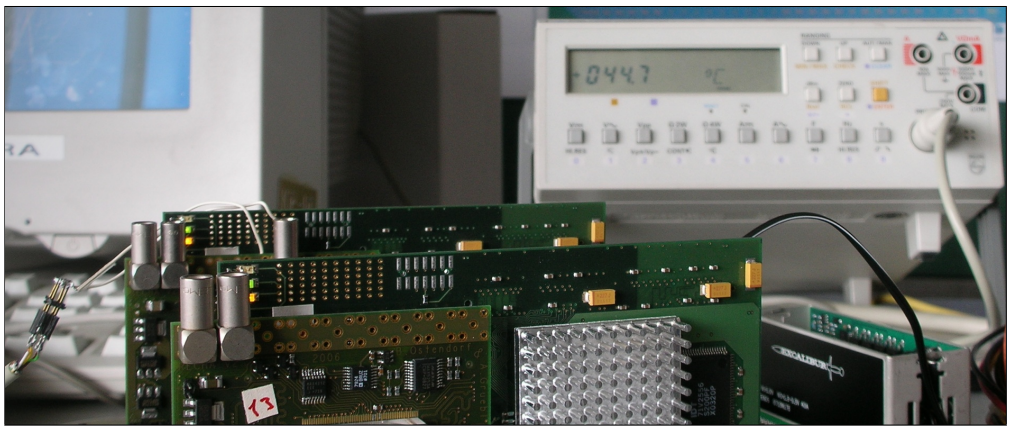


Figure IV.11: *The experimental setup used for measuring the chip temperature.*

Figure IV.11 shows the experimental setup including the multimeter and the cables attached to the probe located between the *Nathan* and the *Recha* board. All measurements involving the chip temperature were taken on *Spikey II, No. 8*.
In order to raise or lower the ambient temperature, the room's window was opened or closed. Occasionally, a hair dryer was used to blow warm air in direction of the hardware from a distance of about one meter.

IV.6.2 Systematic Investigation

First of all, it was observed that the chip temperature rises to high values that might damage the circuitry after repowering the entire backplane and configuring the *Nathan-FPGA*. Until Spikey's so-called *reset flag* has been cleared, the on-chip DAC does not update the voltage and current memories. Thus, the currents drift to high values heating up the chip. Additionally, the communication bus between the FPGA and Spikey's digital part is in `BusIdleMode`, with a high default voltage inducing additional currents in the digital part. The major states of the chip's *analog* and *digital domain* as well as the bus from 'power-on' until 'regular operation' are listed in table IV.1. The compilation of these states was kindly supported by Dr. Andreas Grübl.
As the chip heats up after power-on, it is advisable to clear the reset flag right after initializing the Nathan-FPGA.[15]

[15] This can be achieved by writing the following two commands to all occupied Nathan slots: `dwrite $NATHAN 2 40 1` and `dwrite $NATHAN 2 40 0`.

last action	digital part	analog part	bus
power-on	off	current memories drift	off
darkwing + nathan bitfiles sent, 'dinit -r'	reset flag set	current memories drift	BusIdleMode
clear reset flag	idle	current memories set to 0	BusIdleMode
`Spikey` constructor called	idle, several status registers set	clock on	EventIdleMode
`Spikey::config()` called	idle	currents updated from parameter RAM	EventIdleMode

Table IV.1: *States of Spikey's analog and digital part and the communication bus to the Nathan-FPGA after repowering the backplane.*

In contrast to the states assumed by the analog and digital part after power-up, which drove the chip temperature close to 60 °C, changes in the network configuration and neural activity left almost no thermal trace on the hardware. None of the following setups increased the measured chip temperature, which was at about 35 °C, by more than 0.3 °C:

- analog recording of membrane potentials via the oscilloscope,
- many high synaptic weights w,
- many high control currents (e.g. `drviout`, `drvirise`),
- almost permanent external stimulation ($\frac{\text{runtime}}{\text{idle time}} = \frac{1}{2}$),
- permanent activity in a bursting recurrent network.

The situation changed as the ambiance was warmed or cooled: Figure IV.12 shows the relation between the temperature of the chip T_{chip} and the surrounding air $T_{ambiance}$. Changes in the ambient temperature are followed by almost identical changes in the chip temperature.

In order to determine the impact of the variation of temperature on the hardware behavior, the chip was warmed with a hair dryer. It was found that the chip's activity revealed a serious temperature dependency: An increase of the chip temperature by less than 3 °C caused a drop of the firing rate of about 20%! Figure IV.13 exemplary shows the time course of the membrane voltage of an arbitrary neuron stimulated with Poisson spike trains. The same setup was applied for different chip temperatures. Obviously, the entire voltage course drops, the warmer the circuitry becomes.

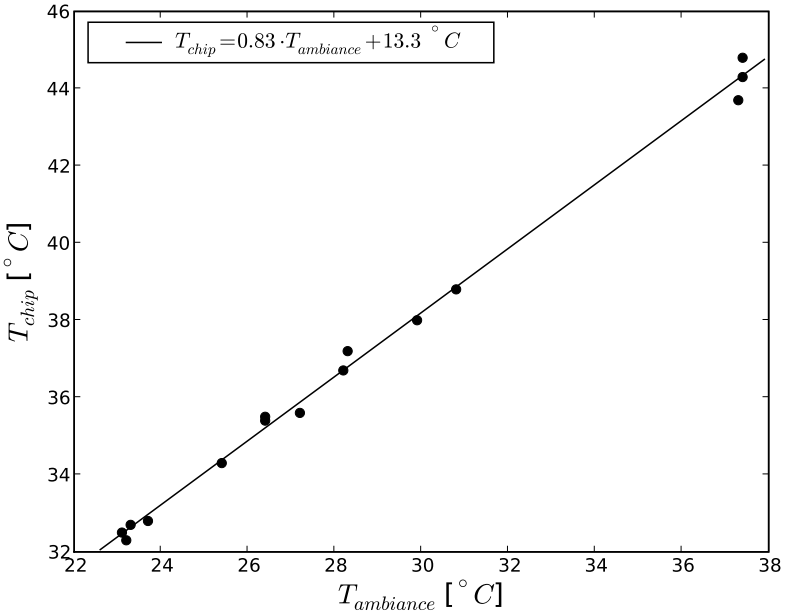


Figure IV.12: *Relation between the temperature of the chip T_{chip} and the surrounding air $T_{ambiance}$. Each point was taken after the ambiance temperature was constant for a couple of minutes. Thus, the chip was supposed to be at thermal equilibrium. The linear function $a \cdot T_{ambiance} + b$ approximating the measurement best is drawn as a black line. The standard errors of the fit parameters, reflecting statistical errors, are $\sigma_a = 0.01$ and $\sigma_b = 0.4$°C. Systematic errors are hard to estimate due to the unknown thermal conductance between chip and probe.*

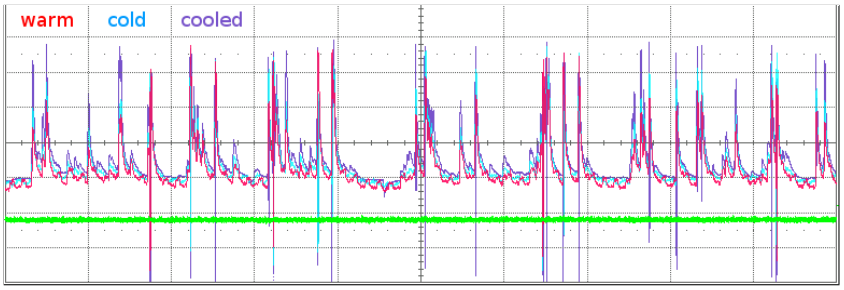


Figure IV.13: *Time course of a neuron's membrane voltage as seen on the oscilloscope (in arbitrary units). The warmer the chip becomes, the more the membrane voltage drops resulting in a lower firing rate. The green trace shows the trigger signal and is of no relevance.*

IV.6.3 Temperature Stabilization

The above results suggest that stabilizing the chip temperature is essential for ensuring reproducible emulation results. As found by [*Müller*, 2008], *Spikey II*'s erroneous voltage drifts of the `vout` voltages lessen as temperature decreases. Therefore, it was favorable to cool down and stabilize the chip at a low temperature. Since *Spikey III* exhibits stable parameters, there is no further need for operating the hardware at low temperature. The following approaches have been tried out to provide a stable thermal setup.

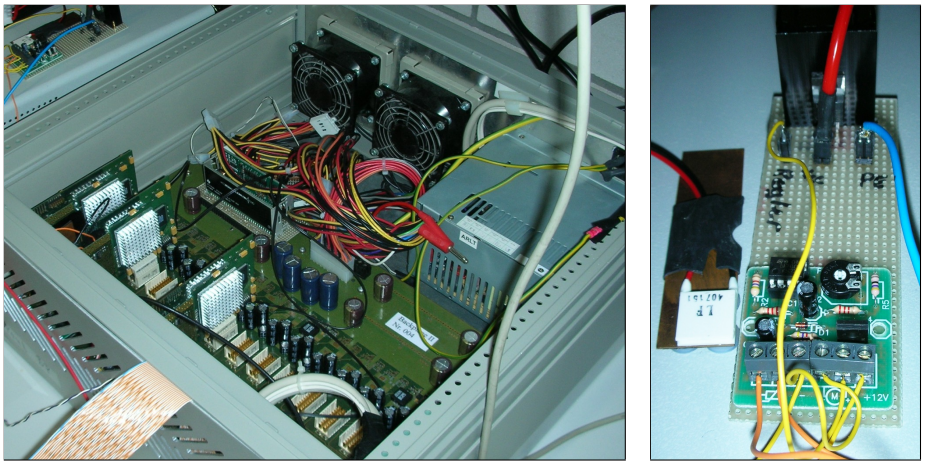


Figure IV.14: left: *The backplane connected to Evolver 13 in a '19-inch rack metal case' (with removed cover). When closed, two fans supply a continuous air stream dissipating the chip's heat. Spikey III no. 25 – the chip used for most experiments presented in this thesis – is located in this case.* right: *The peltier cooling device and its controller.*

With help of Dr. Andreas Grübl the backplane connected to *Evolver 13* was installed into a *19-inch rack metal case*. A photograph of the opened case is shown in figure IV.14 (left). While this method does not support *cooling below the ambiance* at least the heat dissipation works at a steady level. Furthermore, the grounded metal case shields the hardware from external electro-magnetic fields.
An *active cooling device* was build by Maurice Güttler according to plans suggested by Dr. Johannes Schemmel. Basically, a *peltier element* attached to the backside of the *Recha board* cools down the chip to the desired temperature. A controller actuates the peltier element via a thermosensitive resistor. Figure IV.14 (right) shows a photograph of a peltier element and the controller. Due to the limited space at the backside of the Recha board it turned out to be difficult to attach both the sensor and the peltier element such that both have a good thermal conductance to the chip but do not interact directly: When the peltier device began to cool intensely, its backside heated the sensor more than the chip cooled it, leading to an even stronger cooling and causing the cooling system to run at maximum power. Without a systematic tuning of the controller and an elaborate placement of the probe, this approach yields no satisfactory improvement

(see also the following *Comparison Measurement*).
By way of trial, the backplane was also placed in a *climate cabinet* in order to provide a constant and cold ambiance. But the SCSI-connector and the power lines led through a flexible notch in the door yielded serious communication errors. Thus, this setup is also not applicable without any further effort.

IV.6.4 Comparison Measurement

To estimate both the capability of the peltier cooling device and the influence of variations of the temperature on long time scales two benchmarks were carried out: The firing rate of one Spikey III with peltier cooling, one without cooling and one uncooled Spikey II were measured (a) for an entire weekend considering natural thermal fluctuations and (b) for about two hours with artificial variations in temperature.

For that purpose, all three Spikey chips, located on the same backplane, were operated in parallel. On each Spikey, 7 neurons were stimulated with excitatory and inhibitory Poisson spike trains, which – once generated – remained unchanged for the rest of the investigation. All configuration currents and voltages were set to equal values on all chips. Only the discrete weight factors were adjusted such that any neuron exhibited an average firing rate $10\,\mathrm{Hz} < \nu < 35\,\mathrm{Hz}$.

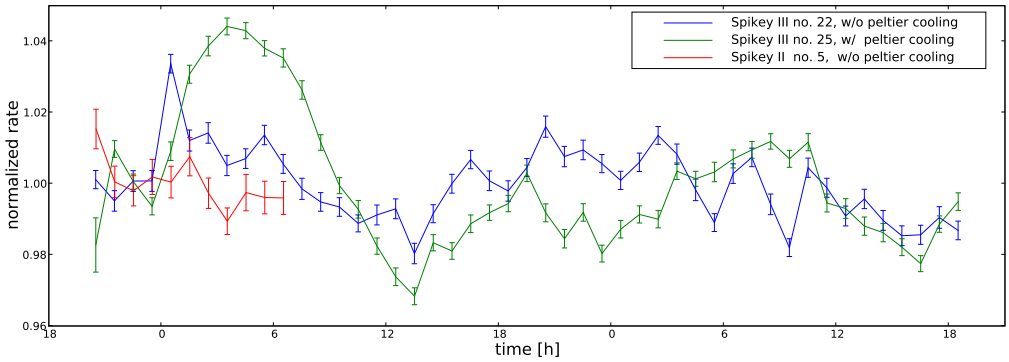


Figure IV.15: *Variation of the chip activity over one weekend in September 2008. All chips were operated by the same backplane and were configured similarly. Especially, the cooled Spikey III chip reveals a different behavior for day- and nighttime. The communication to Spikey 5 broke down after about 12 hours. The error bars reflect the uncertainty of the 60-minutes mean firing rates (see body text). The trace of each Spikey III chip is based on about 60.000 hardware runs, each emulating 4 biological seconds.*

Then, every 30 seconds each chip was run 10 times in a row for 4 biological seconds. Such a sequence of 10 runs is referred to as a *trial* in the following. For each trial the mean firing rate ν of all 7 neurons is stored. Figure IV.15 shows the results over one weekend in September 2008. Every chip's trace has been divided by its mean value normalizing the firing rate to 1. In order to reduce statistical errors an average is made

over groups of 120 trials (corresponding to 60 minutes), as temperature variations are assumed to be negligible over this time span. The error bars reflect the uncertainty of the mean values $\sigma_{<\nu>}$, more precisely: $\sigma_{<\nu>} = \sigma_{\{\nu\}}/\sqrt{120}$. If there exists no source of error acting on long time scales, meaning that all errors are already covered by the 60-minutes uncertainties $\sigma_{<\nu>}$, the standard deviation of the set of mean values $\sigma_{\{<\nu>\}}$ equals the mean value of the 60-minutes standard deviations $< \{\sigma_{<\nu>}\} >$.
With a ratio $\xi := \frac{\sigma_{\{<\nu>\}}}{<\{\sigma_{<\nu>}\}>} = 7.3$ the cooled Spikey III performed worse than the uncooled Spikey III that revealed a lower variation on long time scales ($\xi = 3.7$). Interestingly, the cooled Spikey III tends to increased activity (corresponds to lower temperature) during the night, while during daytime activity drops. The communication to the Spikey II chip broke down after about 12 hours. In general, Spikey II reveals a larger variance inside each trial than the Spikey III chips reflecting that the latest revision of the FACETS Stage 1 Hardware exhibits a higher parameter stability resulting in smaller fluctuations on short time scales.

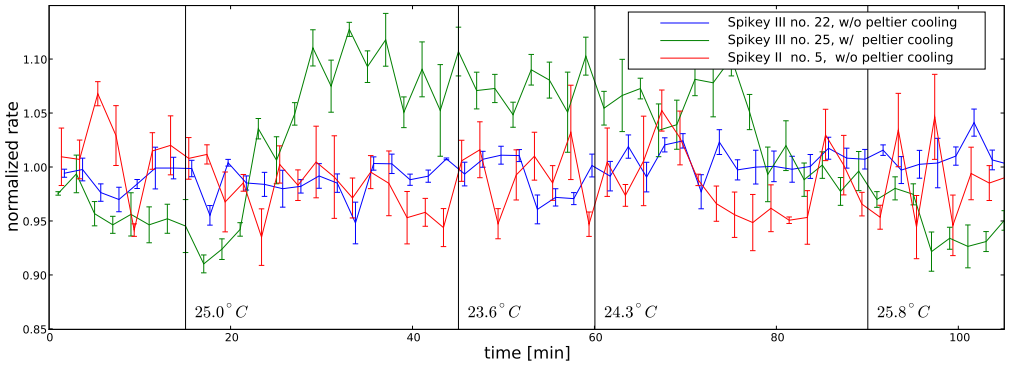


Figure IV.16: *Variation of the chip activity over two hours. The setup is similar to the one presented above, except for each 4 trials are averaged, here. At the beginning, the window of the laboratory was closed. The first vertical line indicates that the window was opened, the second one that it was half open, the third one that it was totally closed. Additionally, the ambiance temperature was taken at these events.*

The same setup was used for a shorter time period in order to have more control over the ambiance. The data acquired as well as the events changing the air temperature are shown in figure IV.16. Apparently, the cooled Spikey III reveals a significant temperature dependency. It has not yet been investigated whether this sensitivity of Spikey 25 arises from the peltier cooling device or is a property of this specific chip.

In summary, it was found that the analog circuity of the FACETS Stage 1 Hardware is sensitive to fluctuations of the chip temperature which exhibits a strong correlation with the ambiance. Hence, it makes sense to stabilize either the hardware directly or at least its surroundings. Multiple approaches serving this purpose have been presented, none of which delivers a satisfactory performance. The additional challenge of cooling

the chip down to a low temperature due to parameter instability has become obsolete with the third revision of the Spikey chip. After a power-on of the system it is suggested to configure the chip with a realistic network setup. Valid data cannot be taken until the hardware is at thermal equilibrium with its ambiance.
For a detailed investigation of the chip behavior a more precise measurement of the chip temperature (e.g. via an IR camera) is indispensable. Furthermore, enhanced control over the ambiance (e.g. via a climate cabinet) must be obtained. A different suggestion for stabilizing the chip behavior is to employ self-stabilizing neural network architectures.

V Software Contributions

Following the credo of the *PyNN*-project to provide portable experiment descriptions (see section II.1), the Electronic Vision(s) Group developed an implementation of the PyNN-API for the FACETS Stage 1 Hardware. The author contributed to maintain this module close to the current state of the PyNN definitions (see section V.1) and added a convenient configuration of hardware synapse dynamics to the front-end (see section V.2). To allow the simulation of the hardware via PCSIM, appropriate neuron and synapse models have either been picked from the 'PCSIM simobject database' or were newly implemented (see section V.3).
In order to allow an interactive operation of the FACETS Stage 1 Hardware, a generic graphical user interface – presented in section V.4 – has been developed. Section V.5 gives an overview of miscellaneous smaller software improvements, contributed by the author.

V.1 Keeping the PyNN-Module up to date

Since the meta-language *PyNN* (see section II.1) is an API under ongoing development, changes in the standard need to be adopted by the back-end specific modules. The author helped to maintain the PyNN-module of the FACETS Stage 1 Hardware and keep it up to date.

The pyNN.ID-Class

All `pyNN.backendName` modules, which implement the back-end specific functionality of the PyNN-API, revert to a set of functions and inherit classes from the common module `pyNN.common`. Among others, this common module provides the functional framework to

- translate back-end specific parameter names to the common standard,
- check whether a requested parameter is existent in the utilized neuron or synapse model,
- ensure correct data types or
- convert values being held in differing units in the back-end.

For the specific back-end, it often suffices to implement fundamental `get()` and `set()` functions that are endowed by the above features with a powerful and convenient functionality. The central class, providing this 'chassis' to the user, is the `pyNN.ID`-class.

When using `pyNN.hardware` every neuron or external input is represented (and managed by the user) as such an `ID`-object. *NEST* and *PCSIM* also apply this concept to synapses.
For instance, when changing a parameter of a neuron created via `n = pyNN.create(...)`, the command `n.v_rest = -70` realizes that this is a valid attribute, converts the voltage from mV to the back-end specific unit and stores the value in the back-end specific memory space.

The author implemented the `pyNN.ID`-class for the `pyNN.hardware.stage1` back-end. As no such unifying data-structure existed before, all functions – as `create`, `connect` or `record` – had to be attuned to the common class. As well, the compatibility to the Hardware Abstraction Layer (HAL) had to be ensured.

The pyNN.Population-Class

Commonly, sets of neurons, featuring similar characteristics, are combined to so-called *populations* when it comes to the simulation of larger networks. The PyNN-API defines a common `pyNN.Population`-class which facilitates the management of such sets. For instance, a `Population p` allows to arrange the contained neurons on two- or three-dimensional grids and to address them via their coordinates `p[m,n]`. Furthermore, it provides functions for iteration, convenient recording or assignment of parameters to all members.
The author implemented `pyNN.Population`s for the `pyNN.hardware.stage1` back-end.

Synaptic connections between `Population`s are established via so-called `Projection`s. This functionality was added to the `pyNN.hardware.stage1` module by Andreas Bauer.

V.2 Configuration of Short Term Plasticity via PyNN

Since neural network experiments on the Facets Hardware are scripted in *python* and *PyNN* a convenient configuration of the synapse dynamics shall be available on this software level. The *PyNN-API* does not yet define an interface to set up short term plasticity via `pyNN.connect()`. Hence, a hardware specific implementation was needed. After consulting the PyNN-developer Andrew Davison the optional dictionary `STP` was added to the `pyNN.connect()`-function. This implementation can easily be modified as soon as PyNN defines a common standard.

V.2.1 Usage and Parameters

The STP-mechanism is part of each synapse driver. Nevertheless, some voltages are shared by multiple synapse drivers: V_{fac}, V_{stdf} and V_{dtc} can be set (in Volts) using `pyNN.set_STP_globals()`. To read the current settings, `pyNN.get_STP_globals()` can be used.
Individual parameters can be passed when establishing a synapse via `pyNN.connect()`

providing a dictionary named `STP`. Its default value is `None`, denoting a *static* synapse. A *dynamic* connection is generated if `STP` holds the following *keys*:

- `'mode' = 'fac'` or `'dep'`: Whether the synapse driver emulates *depressing* or *facilitating* synapses.
- `'cap' = 1, 3, 5, 7`: The change of the (in-)active partition after each spike can be controlled by setting the capacitance C_2 in units of $C_1/8$. The allowed values reflect the possible hardware settings.

As each synapse driver can emulate only one synapse type at a time, network topologies given as a *PyNN*-script might be contradicting. E.g. a driver can not serve *depressing* and *facilitating* synapses, simultaneously. In that case, an exception will be raised.

At present, plasticity parameters are hardware specific, e.g. the recovery time constant is passed as a voltage. This is due to the fact that the hardware voltages have not yet been mapped to their biological counterparts. A proposal for measuring these inner parameters with indirect techniques is presented in section IV.5. An implementation of more generic parameters is highly desirable.

V.2.2 Details of the Software Structure

When instantiating a dynamic synapse in PyNN, this information needs to be carried into the low level structures of the code and translated into data suitable for the FACETS Stage 1 Hardware back-end. This section outlines the information flow for short term plasticity. In a similar manner STDP configuration data could be handled. Therefore, a con-generic STDP-data-structure without specific functionality has been implemented.

`__init__`.py and pyhal_s1v2.py: The dictionary holding the STP-configuration simply gets passed through `pyNN.connect()` and `pyNN.hardware.connect()` to the hardware specific connect command of the *Pyhal Building Blocks*.
Global parameters, set via `pyNN.set_STP_globals()`, are written to the corresponding voltages in the `HWAccess`-object. The `pyNN._synapsesChanged` flag indicates that this information is outdated in *Spikey's parameter RAM* and needs to be retransmitted.

pyhal_buildingblocks_s1v2.py: The `Network`-Object holds a list of `Neuron`s. These keep track of their incoming and outgoing synapses. The synapses no longer are simple `float`s, but the newly introduced `Synapse`-class inherits from `float`. Beyond the weight w of static synapses, the class also provides STP- and STDP-data for the *mapping process*.

pyhal_config_s1v2.py: The `HWAccess` member function `mapNetworkToHardware()` calls `synapseStatusByte()` when preparing the `SpikeyConfig` object. The latter function extracts the weight- and STP-information from the `Synapse`s and translates it into the synapse driver's *status byte* presented in section I.3.

Furthermore, `mapNetworkToHardware()` passes the global STP-voltages to the `SpikeyConfig`-object.

pyspikeyconfig.cpp: The *python-wrapper* to access the `SpikeyConfig`-object. Beside providing read-/write-access to the `SpikeyConfig` member variables, it checks for ambiguous synapse driver configuration.

When the `SpikeyConfig`-object holds the desired configuration and all *update flags* are set properly, the low level software structures will transmit the data to the Hardware.

V.3 Implementation of the FACETS Hardware Model in PCSIM

In order to provide a software simulator back-end revealing the same network dynamics as the FACETS Stage 1 Hardware, Spikey's neuron and synapse models have been implemented in PCSIM [*Pecevski and Natschläger*, 2008]. This simulator is being developed under administration of Dejan Pecevski[1] and Thomas Natschlaeger[2] of the FACETS member TU Graz. The hardware model was implemented by the PCSIM developer DI Klaus Schuch and the author.

The usage of a software simulator can facilitate the work with a hardware system, as it provides and improves necessary understanding:

- The software simulator gives knowledge about and precise control over most internal parameters, like the time course of synaptic conductances or the variation of resting potentials over a population of neurons. This supplies a great aid on searching for appropriate network parameters.

- Since variations of hardware components due to the production process might be unknown or not yet been measured, a software simulator allows to distinguish inherent network properties from hardware specific effects. Thus, the extent of hardware inaccuracy can be rated from a network point of view.

- After the behavior of the chip and the simulator have been adjusted to a sufficient degree, the hardware emulation provides a fast back-end for parameter sweeps or long term experiments.

In the following the recommended PCSIM objects to use in conjunction with the Hardware are presented. Some objects are common models, others have been tailored to the chip.

[1] dejan@igi.tugraz.at
[2] tnatschl@igi.tugraz.at

Neuron Model The PCSIM simulator provides a conductance based leaky integrate and fire neuron (class `CbLifNeuron`) with the same underlying differential equation as the hardware neuron.

Basic Synapse Model For static synapses the `StaticCondExpSynapse` is an adequate approximation to the hardware synapse. When triggered it rises to it's maximum conductance value instantaneously and decays exponentially. The maximum weight does not vary over time.

Dynamic Synapse Model In order to simulate the Short Term Plasticity mechanism of the FACETS Stage 1 Hardware, two new synapse classes – namely, `FacetsHWDepressionExpSynapse` and `FacetsHWFacilitationExpSynapse` – have been added to the 'PCSIM simobjects database'. Like the `StaticCondExpSynapse` they attain their maximum conductance values without delay, but here the maximum weights change over time. More precisely, they exhibit the hardware synapse dynamics in *depression mode* and *facilitation mode*, respectively. The software variables and the corresponding hardware parameters are listed in table V.1. For the synapse driver circuits and the naming of their components see section I.3.

PCSIM			
Depression	**Facilitation**	**Hardware**	**comment**
max_dep	max_fac	$\mu \cdot V_{stdf}$	scale factor for strength of STP
–	norm	V_{fac}/V_{stdf}	facilitation reference voltage
inact_tau	act_tau	V_{dtc}	recovery time constant
inact_step	act_step	$C_2/(C_1 + C_2)$	relative step after each spike
inact0	act0	always ground ($\hat{=} 0$)	initial values of the (in-)active partition
W	W	$w \cdot f(\texttt{drviout})$	synaptic weight without STP modification

Table V.1: *The parameters describing short term plasticity in software and hardware. The hardware mechanism is discussed in section I.3.*

V.4 Instant Interaction via a Universal GUI

In order to provide a convenient interaction with experiments, the author developed a universal graphical user interface (GUI), called the *borescope*. It can help to gain a qualitative understanding of interrelated parameters by interactively investigating the influence of different configurations on a measured variable, or to search roughly for proper parameter ranges. Above all it allows an intuitive configuration with instant readout of the hardware at the same time.

The diversity of possible setups and types of measurands demands a very generic software structure from the borescope. Furthermore, it must provide an intuitive user interface and needs to be 'plugged' easily to the parameters of interest.

Currently, the borescope features:

- 3 programmable sliders (providing both step-less and discrete values)
- 2 binary check-boxes (on / off)
- 2 graphical outputs
- 2 textual outputs

Any feature can be connected to any value accessible to the probed experiment. Free programmable functions are executed, whenever a controller element changes its value. A screen-shot of the borescope is shown in figure V.1. A snippet of code demonstrating the configuration is presented in table V.2.

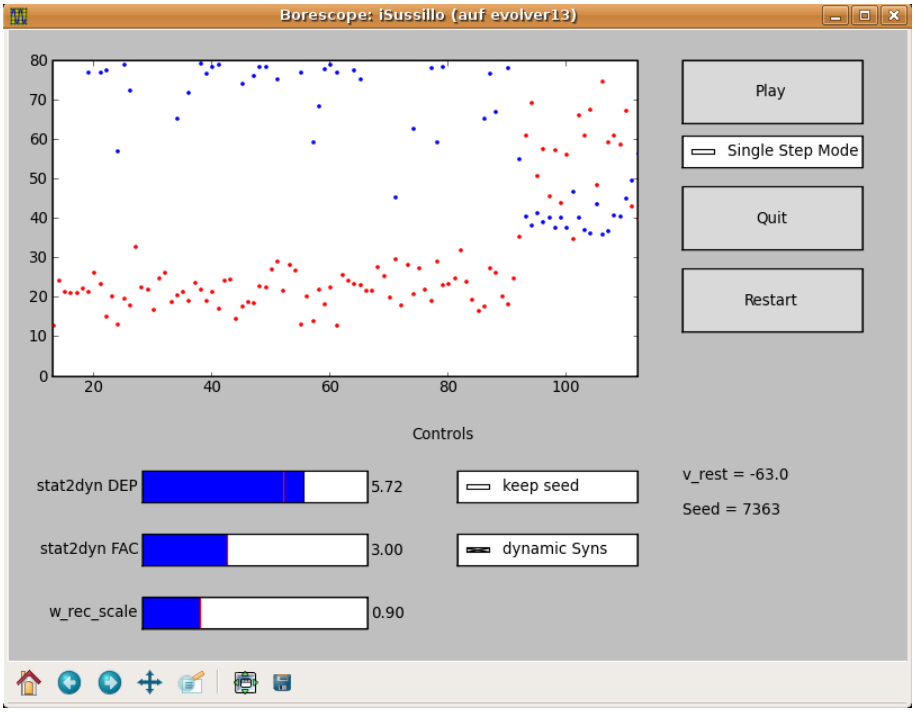


Figure V.1: *A screenshot of the Borescope. The example shows an instant interaction with the network architecture presented in section III.3.*

Generally, parts of the plugged script are executed in a loop. The loop can be paused manually, or automatically after every pass. Other parts of the script serve for initialization. To provide this information on different parts, an experiment must be provided as `Experiment`-Class, consisting of a `onLoad()`-, a `prepare()`- and a `go()`-function. Since almost every experiment (logically) satisfies this structure, the integration of any experiment into the borescope should be an easy task. On the other hand, every script, written for the borescope, can also be executed solely – it is not dependent on the GUI. The definition of parameters to be monitored and functions to be executed is done via an additional configuration file.

A documentation of the usage, including useful examples and details on the execution flow, can be found in the borescope's directory, filenames: `readme.txt`, `experiment_template.py` and `experiment_template_cfg.py`. The borescope was used regularly by the author, while this work was done.

```
cfg.slider.1 = [ 'stat2dyn DEP' , 5 , 0 , 8 , False ]

def prepare.slider.1(self):
    self.stat2dynFactor['dep'] = self.controlling.GUI.slider.1.val
    self.newNetwork = True
```

Table V.2: *Code demonstrating the configuration of a slider for the borescope. The first line defines appearance and ranges of the slider. The function defined from the second line of code determines the functionality of the slider. The example is taken from the `_cfg.py`-file of the experiment shown in figure V.1.*

V.5 Other Software Improvements

The author helped to improve and maintain the software layers presented in section II.1. The usage of some useful features is described in the following.

Restriction to Recordable Neurons

As stated in section I.4, only a small subset of neurons can be recorded reliably on *Spikey III*. The exact number of such 'recordable neurons' varies from chip to chip. The author added an attribute to the workstation setup file `workstations.xml`, storing the list of neurons that were found to be recordable for each workstation. The `pyNN.record()`-function evaluates this information and prints a warning, if errors during readout are likely to occur.

Assigning Workstations to Users

Since one backplane usually operates multiple workstations, a user might accidentally send data to a chip currently emulating networks of someone else. In order to 'protect' a user's workstation from erroneous reconfiguration, the author added a warning that is displayed if a chip is about to be utilized, which was not marked to be the user's default workstation via `workstations.mine`.

Automated Spike File Generation

In contrast to software simulations, which typically reconfigure the network after every run, it is common practice for hardware emulations to rerun the same setup several times in order to estimate or reduce the influence of analog noise. An option was added to the `pyNN.hardware.stage1` module that allows to conveniently store the results of every run in distinct files. It is activated by passing an argument to the initialization routine: `pyNN.setup(..., incrementFilename=True)`.

pyNNgroups

As not all neurons are reliably recordable simultaneously, neurons of interest need to be mapped to specified locations on the chip. In the future, this task will be performed automatically, using the graph model presented in section II.3. A small preliminary mapping tool – `pyNNGroups` – was implemented *on top of PyNN* by the author. It was primarily developed to conveniently map a synfire chain to the chip (see section VI.1) and adopts some features of the `pyNN.Population`-class, which was not available to the hardware module at that time.

VI Realizing Network Paradigms on the Hardware

The development of the FACETS Hardware aims for the emulation of neural networks, which possess computational power or other types of functionality, like activity modulation or signal propagation. This chapter presents the implementation of three concepts that provide or improve information processing power for the underlying substrate.
As presented in section VI.1, a stable closed-loop synfire chain, see sectionIII.1, was successfully emulated on a Spikey chip. Synfire chains feature strong feed-forward connections, with a high tolerance to variations in the synaptic properties. Other network architectures require a more precise adjustability and a rather homogeneous substrate. It is common practice to expose these networks to a permanent background conductance, modeling a biologically realistic high-conductance state (see section III.2).

The inhomogeneities of the current revision of the FACETS Stage 1 Hardware exceed an acceptable level, as shown in chapter IV. Furthermore, only few input channels – with limited input bandwidth – are available, impeding an independent spike-based stimulation of all neurons throughout the network. In section VI.2, an alternative concept of stimulation is presented, which allows the approximation of a high-conductance state under the above constraints. Finally, a self-stabilizing network architecture is investigated regarding its applicability to the hardware. It will be found that such networks are capable of reliably counterbalancing inherent variations of hardware components (see section VI.3).

VI.1 Synfire Chains

The *synfire chains* presented in section III.1 reveal a rather simple architecture. Additionally, a synfire chain's functionality can be assigned to three basic types:

- The chain aborts: Even if the first layer is strongly stimulated the chain is not able to keep up activity across all layers.
- Activity disperses: The time span a single layer fires increases with each layer. In case of closed-loop synfire chains this results in permanent activity.
- The chain works as intended: The pulse packets stay focused and activity does not break down.

Due to these clearly separated states it is likely that a stable synfire chain can be established even if all network parameters are subject to quantitative errors. This raises hope

to successfully emulate synfire chains on an imperfect analog substrate like the FACETS Stage 1 Hardware.
However, the topology of the present hardware system (see section I.5) restricts the maximum size of a single chain to 192 neurons, while successful simulations, as carried out by the authors of [**?**], used 3750 neurons per synfire chain.[1] Hence, a balance between wide layers (increasing stability by outweighing inhomogeneities) and long chains (reducing negative effects of dispersion due to lower frequencies in case of closed-loop chains) must be found. Supportingly, dispersion can be counteracted by adding sparse, random recurrent inhibitory connections: Strong activity in one layer damps the activity of other layers. This concept, used by [**?**], was adopted for the hardware emulation.

A synfire chain was set up using the `pyNNgroups`-class (see section V.5) as it provides methods to conveniently map selected neurons in each layer to recordable hardware neurons (see section I.4). The search for suitable parameters – weights, connectivity probabilities, chain length and layer width – was performed by interactive execution with the *borescope* (see section V.4).

The spike pattern of a looped-back chain emulated on a *Spikey III* chip is shown

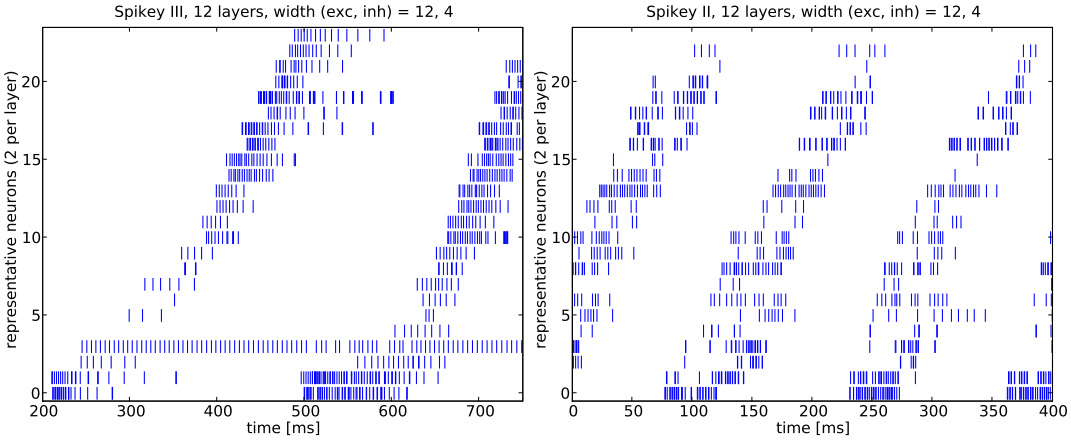


Figure VI.1: *Looped-back synfire chains on the FACETS Stage 1 Hardware.* left: *a chain on Spikey III initiated via an external excitatory pulse at* $t = 200\,\text{ms}$. right: *a stable synfire chain on Spikey II – the initial stimulus was injected minutes ago.*

in left panel of figure VI.1. The spikes of one excitatory and one inhibitory neuron per layer were recorded. The chain was stimulated by injecting a strong excitatory input pulse (duration: 10 ms) into the first layer at $t = 200\,\text{ms}$. After about 300 ms, activity has passed the entire chain and the last layer excites the first one again. Due to variations from chip to chip concerning synaptic strength and neural excitability, parameters need to be adapted for each system individually. For the same reason there is no benefit in

[1] Personal communication with Sven Schrader

exploring parameter spaces separating the different types of functionality, as the results would solely be valid for a single chip.
The right panel of figure VI.1 shows an extract of a remarkably stable chain run on a *Spikey II* chip: The chain was initiated during another run minutes before the plotted data was recorded. This example demonstrates the fundamental difference between hardware emulations and software simulations: As long as the chip is not being reconfigured, it will follow the network dynamics – independent of external control. Accordingly, a 'hardware experiment' does not necessarily involve a reconfiguration of network parameters. In case of the plotted synfire chain, it simply reads back the spikes of several neurons for a preset duration.
Some looped-back synfire chains on this Spikey II chip were observed to be still stable after more than 30 minutes *in real time*, corresponding to more than 2 years in the biological interpretation.[2]

Although each Spikey chip contains $2 \cdot 192$ neurons, thus providing enough neurons for two synfire chains, restrictions to the network topology (see section I.5) prevent a generic implementation of interconnections, which could have possibly allowed for the chains to bind as described in section III.1. A laborious manual design and mapping of such a network has not yet been performed. The upcoming *multi chip operation* of the FACETS Stage 1 Hardware will provide a substrate supporting multiple interconnected generic synfire chains.

VI.2 An Approximation for a High Conductance State

As outlined in section III.2 cortical neurons in alive animals often experience a permanent synaptic input which – presumably – does not carry any particular information, but fundamentally changes neural spike processing. In computer simulations this background input can be modeled as an additional conductance between the membrane and the different reversal potentials which is drawn from a normal distribution (with non-zero mean) at each simulation step.
The FACETS Stage 1 Hardware does not provide a similar mechanism. A generic approach for modeling *high conductance states* would be stimulation via Poisson spike trains. But since each neuron needs to be excited with a high input rate and low synaptic weights in order to achieve the intended mean and standard deviation of the background conductance, this concept poses several problems:

- All spike trains must be generated by the software and then be transmitted to the chip, slowing down the operation of the hardware due to the configuration overhead.
- Larger networks might only provide 64 external input channels for both *experiment specific* and *Poisson* input (see section I.5). This easily results in a shortage of available inputs and bandwidth.

[2] with a speedup factor of $5 \cdot 10^4$

- If there exist more neurons than input channels, the background stimulation of the neurons will not be pairwise independent. This might cause parasitic network correlations distorting the emulation results.

The author introduced an alternative for modeling high conductance states on Spikey reducing the above issues, which is presented in the following. The conductance towards each reversal potential caused by the background stimulation, as applied in computer simulations, can be expressed as a constant mean value endued with moderate noise $g(t) = \bar{g}_s + g_s(t)$, as long as background activity remains statistically unchanged during the considered time span. The mean of $g_s(t)$ vanishes over time. Regarding the sub-threshold dynamics of a *leaky integrate-and-fire neuron* (see section I.2) with – for the sake of simplicity – only one synaptic reversal potential V_{rev}, the mean background conductance $\bar{g}_s$ just alters the *effective resting potential* $\tilde{V}$ and the *effective membrane time constant* $\tilde{\tau}$:

$$\begin{aligned}
-C_m \frac{\partial V_m}{\partial t} &= g_{leak} \cdot (V_m - V_{rest}) + g(t) \cdot (V_m - V_{rev}) \\
&= g_{leak} \cdot (V_m - V_{rest}) + \bar{g}_s \cdot (V_m - V_{rev}) + \text{noise} \\
&= (g_{leak} + \bar{g}_s) \cdot \left[V_m - \left(V_{rest} + \frac{\bar{g}_s}{g_{leak} + \bar{g}_s} \cdot (V_{rev} - V_{rest}) \right) \right] + \text{noise} \\
&=: \frac{C_m}{\tilde{\tau}} \cdot (V_m - \tilde{V}) + \text{noise}
\end{aligned}$$

This equation can easily be expanded to multiple conductance channels with different reversal potentials. The sub-threshold dynamics always result in an effective reversal potential $\tilde{V}$, an effective membrane time constant $\tilde{\tau}$ and additional noise on every ion channel.

Hence, a high conductance state can be modeled by adjusting the *resting potential* V_{rest} and the *membrane leakage conductance* g_{leak} properly accompanied by moderate noise generated with Poisson spike trains. Regarding the FACETS Stage 1 Hardware these *noise inputs* can be operated with low firing rates and higher synaptic weights. Thus, each neuron only occupies few background channels. By connecting each neuron to a small subset randomly drawn from the noise inputs, erroneous network correlation caused by background stimuli can be reduced considerably.

Furthermore, this approach considers a dis-proportionality of *Spikey III's* intrinsic time constants: Synaptic time constants τ_{syn} typically exceed membrane time constants of unstimulated neurons $\tau_{leak} = C_m / g_{leak}$ by a factor 3. In case of high conductance states, this ratio between τ_{syn} and $\tilde{\tau}$ is biologically realistic. Hence, only V_{rest} needs to be adjusted when modeling high conductance states on Spikey III.

The network presented in section VI.3 makes use of this technique.

VI.3 Self-tuning through Short Term Plasticity

In view of the inaccuracies and instabilities presented in section I.5 and chapter IV an adequate predictability of the emulated network behavior can only be achieved by providing a *substrate on the network level* that compensates for chip-inherent variations. An auspicious and rather universal approach was presented in section III.3 using short term plasticity. As the computer simulations carried out by [*Sussillo et al.*, 2007] were based upon larger networks, exceeding Spikey's capacity by more than one order of magnitude, it remained unknown whether this paradigm is applicable to the hardware. This section addresses this issue.

In the first part, it is investigated how such network architectures can be mapped to the chip, while optimizing the hardware efficiency, and how external stimulation can be applied such that little correlation is evoked. Following these considerations, the implementation in PCSIM and the results of the computer simulations carried out by Klaus Schuch[3] and the author under the supervision of Prof. Dr. Wolfgang Maass[4] during a visit at the FACETS member TU Graz are presented. As the basic properties of self-stabilizing network architectures have been found to be fulfilled in the simulations even for small networks, the concept was applied to the FACETS Stage 1 Hardware. Even though the emulations proved elementary features of self-stabilizing networks to be met on the hardware, not all integral network properties have been investigated so far.

VI.3.1 Experimental Setup and Applied Measures

As shown in figure III.1, any neuron must be capable of projecting on neurons in both populations simultaneously, whereas the dynamics of the synaptic connections depend on both the type of the pre-synaptic and the post-synaptic neurons. As stated in section I.5, any synapse driver of the FACETS Stage 1 hardware only supports one `mode` of plasticity at a time. Furthermore, the index of a synapse driver confines the choice of its pre-synaptic source. Considering these **topological restrictions**, the largest generic[5] network mappable onto the chip comprises 192 neurons. A possible partitioning of the resources is shown in figure VI.2.
Obviously, any required *recurrent synaptic connection* can be established with this setup. But since there only exist 256 synapse drivers per synapse array, 192 of which are used for recurrent connections, 64 independent inputs must suffice to provide **external stimuli** to all 192 neurons.[6] Furthermore, the maximum *input bandwidth* between Spikey and the playback memory on the Nathan board is limited as stated in section I.4. This bottleneck narrows the maximum average rate of 2×64 external Poisson inputs to about

[3] schuch@igi.tugraz.at
[4] maass@igi.tugraz.at
[5] A network connectivity based on probability distributions rather than connections 'designed by hand'.
[6] The current software HAL does not allow for projecting different external input spike trains onto the network via synapse drivers of the same sub-index located at different synapse arrays. Hence, the input of the left array can only be mirrored to the right array.

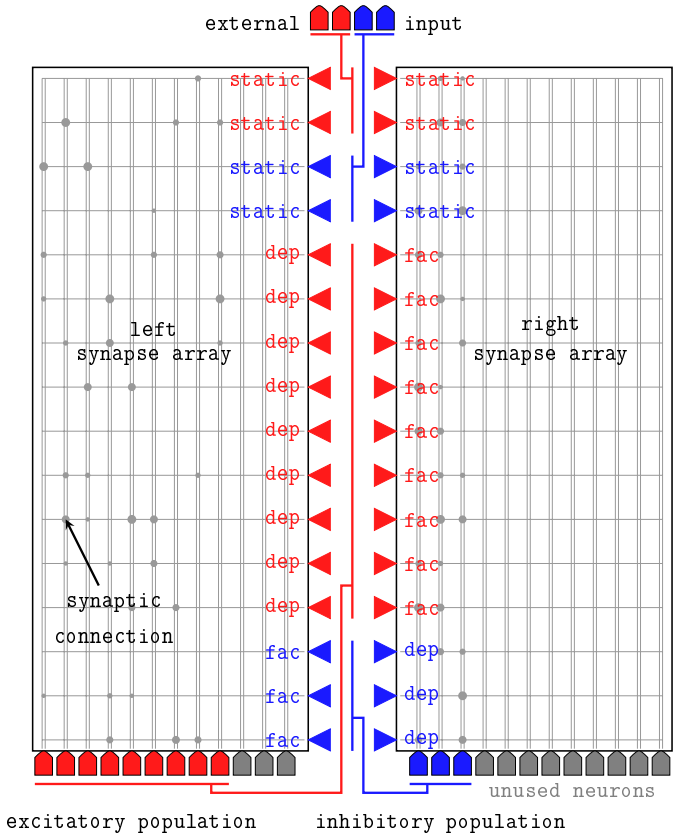


Figure VI.2: *Mapping of the recurrent network architecture outlined in figure III.1 onto the FACETS Stage 1 Hardware. The largest randomly connected networks supported by the chip comprise 192 neurons, leaving 2×64 synapse drivers for external stimulation. The software simulations are supposed to consider all known hardware constraints.*

15 Hz per channel[7].

Nevertheless, external stimulation ought to be uncorrelated (ideally, individual Poisson spike trains for each neuron), even if each neuron requires a high total pre-synaptic firing rate to rise above its threshold voltage V_{thresh}, when realistic synaptic weights are used. In order to reduce input driven network correlation, each neuron receives input from a small randomly chosen subset[8] consisting of both excitatory and inhibitory inputs. Additionally, the neuron resting potential V_{rest} is set to a higher voltage than in case of a 'resting state', while its membrane time constant $\tau_{mem} = C_{mem}/g_{leak}$ is reduced. This method of modeling stimuli does not disturb the dynamics of conductance-based leaky integrate-and-fire neurons and is capable of approximating a high-frequent conductive input stream as presented in section VI.2. Accordingly, the mean value and the variance of external stimulation will be identified with the resting potential V_{rest} and the synaptic weight of the connected inputs w_{inp} in the following. In any setup, static synapses are used for external stimulation.

25% of the **neurons** are chosen to be inhibitory, yielding 144 excitatory and 48 inhibitory neurons. All neurons possess the same mean parameters listed in table VI.1. These mean values either are spread by drawing them from a probability distribution or are subject to inherent hardware fluctuations.

Regarding the **recurrent connections**, two scaling parameters are utilized:

- a *global connectivity multiplier* c_{scale} affecting the connection probability between any two neurons and
- a *global recurrent weight multiplier* w_{rec} adjusting the synaptic weight between any two neurons.

These scaling parameters are multiplied with a base value of the **connection probability** and the synaptic weight determined by the type of the pre- and post-synaptic neuron. Altogether, there exist four types of connections: $e \to e$, $i \to i$, $e \to i$, $i \to e$, with the first pair denoting intra-connections and the latter pair denoting cross-connections.[9] The base values of the connection probabilities are set such that (a) any neuron exhibits as many excitatory as inhibitory incoming recurrent connections and (b) neurons of the inhibitory population receive twice as many incoming connections as excitatory neurons. Although this rule does not rely on biological evidence, it was found to be suitable for the considered architecture. It follows the idea that the moderating inhibitory neurons ought to sense the averaged activities of the entire network and that the influence of a population on the network can – as a first approximation – be controlled via the synaptic

[7] with `speedup` $= 10^5$

[8] Typically, about 8 out of 64 in case of hardware emulations and 5 out of 48 in case of software simulations.

[9] Excitatory neurons are marked as e, inhibitory neurons as i.

parameter	mean	spread PCSIM	spread & noise Spikey
C_m [nF]	0.2	0.3 / 0.5	exact per definition
R_m [MΩ]	25	1.0 / 0.8	–
g_{leak} [nS]	40	–	1.0 / 0.6
V_{thresh} [mV]	-55	0.01 / 0.02	see section IV.4
V_{reset} [mV]	-80	0.1 / 0.2	see section IV.4
$V_{rev,exc}$ [mV]	0	0	unknown (unstable)
$V_{rev,inh}$ [mV]	-80	0	≈ 0
τ_{refrac} [ms]	1	0	see section IV.4
τ_{syn} [ms]	30	0.3 / 0.5	≈ 10 ms

Table VI.1: *Neuron parameters used for the self-stabilizing network architectures. The spread is given relative to the corresponding mean in the form* relative standard deviation / relative bound. *If the value, drawn from a normal distribution, exceeds the bounds, it is redrawn form a uniform distribution within the bounds. The PCSIM neuron model uses* R_m $(= g_{leak}^{-1})$ *for defining the membrane leakage conductance. The mean values yield* $\tau_{mem} = C_m \cdot R_m = C_m / g_{leak} = 5\,\text{ms}$*, reflecting the approximated high-conductance state. Regarding the hardware, the variation of some parameters is unknown. Recently, it was found that the excitatory reversal potential becomes unstable under high load, e.g. in case of high network activity. Variations in* C_m *are compensated via a calibration of* g_{leak}*, but still affect the size of EPSPs as the 'weight transformation calibration' presented in section II.2 was not used. The current Spikey revision lacks adjustability of* τ_{syn}*, the falling time-constant of the exponential synapse.*

weights. For given population sizes, this rule determines the base values of the connection probabilities (while preserving c_{scale} as a free parameter).[10]

The base values of **recurrent synaptic weights** refer to the weight of static synapses and are chosen differently for the software simulation and the hardware emulation. How the choice of these values affects the network behavior, and to what amount 'good' parameters depend on the specific Spikey chip used, should be subject to further studies. In case of dynamic synapses, the short term plasticity mechanism presented in section I.3 and analyzed in section IV.5 is used. In order to provide the software simulator with this plasticity mechanism, a synapse model of the *FACETS Hardware Synapse* was implemented in PCSIM (see section V.3). The *weight conversion* between dynamic and static synapses is determined under an equidistant input of constant firing rate ν_{target} as presented in section IV.5.

So, how to **measure the performance** of such a network architecture in general, and especially the influence of dynamic synapses?

Regarding the average firing rate of the entire network with respect to different amounts of external stimulation (see above), a *self-tuning recurrent network* should increase its activity compared to a purely *input driven network* ($w_{rec} = 0$) for weak stimuli, while under strong input, yielding a high firing rate in case of input driven networks, activity should be moderated. Ideally, the total firing rate exhibits a plateau on a medium level[11] over a wide range of external stimulation.
Provided that the network is capable of tuning its average firing rate, a closer look can be taken at the firing rates of the two populations ν_e and ν_i. While a homogeneously stimulated input driven network yields an equal activity of both populations $\nu_e - \nu_i = 0$, a self-tuning network is expected to exhibit a higher-than-average excitatory rate $\nu_e - \nu_i > 0$ when adjusting its activity upwards, and a higher-than-average inhibitory rate $\nu_e - \nu_i < 0$ when moderating strong external excitation.
Since for the construction of a network only probability distributions and connection probabilities are provided, an individual network might exhibit specific and non-typical properties. Thus, regarding the above measures, only the activity averaged over a multitude of networks, generated from the same probability distributions, is taken into account.
Having said this, also a different meaning of 'self-stabilization' should be considered: To what extend will the properties of different networks, generated from the same probability distributions, vary? This issue touches the predictability of generated networks and can be quantified by considering not only the activity averaged over multiple entities, but also the standard deviation of the mean firing rates of different networks.

[10] Resulting base values of connectivity: $c_{e\to e} = 4$, $c_{i\to i} = 24$, $c_{e\to i} = 8$, $c_{i\to e} = 12$. The connection probability between any two neurons is obtained by multiplying the respective base value with the scaling factor c_{scale}.
[11] Referring to section III.3, a rate of about $5\,\mathrm{Hz} < \nu < 20\,\mathrm{Hz}$ can be assumed as a *medium level*.

A network architecture featuring both of the above meanings of self-stabilization may be applicable in order to *reliably counterbalance hardware specific variations* and thus, provide a *substrate on the network level* for ambitious computational tasks, like those applying the concept of liquid computing.

As stated above, the external Poisson input spike trains are intended to stimulate the network such, that as little correlation, caused by the small number of independent channels, as possible is transferred to the neurons. But this does not guarantee that a recurrent network will not synchronize. In fact, it can be suspected that a network, being capable of self-tuning its average firing rate, tends to correlate its activity. Such correlation, not induced by the input stream, could impair computational power as processed information might be overwritten by internal dynamics. Hence, network correlation should be taken into account when investigating recurrent network architectures.

In the following, the above concepts are applied to both purely input driven (i.e. $w_{rec} = 0$) and recurrent networks, while for the latter the performance of static and dynamic synapses is compared. All setups are realized in both PCSIM and hardware, with certain differences in the configuration that turned out to be indispensable.

VI.3.2 Implementation in PCSIM

The implementation and simulation of the described network architecture was performed with assistance of Klaus Schuch during a six-week visit at the FACETS member TU Graz in May / June 2008. As the *pypcsim* software layer provides additional modules for setting up and analyzing neural networks, all scripts were written in this python-based, but back-end-specific, programming language.
The neuron and synapse models suggested in section V.3 were used, which have partly been written for simulating the dynamics of the FACETS Stage 1 Hardware.[12] All calculations simulated 5 sec of biological time with a simulation time step of 0.1 msec. For measuring both firing rates and network correlation, only the last 4 sec of each trial were taken into account, since the network needs some hundreds of msec for leveling out.
The applied short term plasticity parameters (cf. section I.3, section V.3 and section IV.5) and synaptic weights are listed in table VI.2.
The network was stimulated with 48 Poisson spike trains (24 exc. & 24 inh.) with rates drawn uniformly from the interval $[8\,\text{Hz}, 12\,\text{Hz}]$. Each neuron received input from $1-3$ excitatory and $1-3$ inhibitory spike sources. In order to model analog noise on the neuron membrane that is not covered by the spread of other parameters, each neuron was subject to an additional current input randomly drawn from a normal distribution in every simulation time step. In accordance to the measured noise level of the hardware, its standard deviation was set to $I_{noise} = 5\,\text{pA}$.

[12] For the sake of simplicity, simple exponential synapses were used not modelling the exponentially rising edge. As the rise time is very short in the synapse drivers, such an approximation is feasible.

parameter	mean	spread
depression		
max_dep	0.3	0.5 / 0.5
tau [ms]	50	0.3 / 0.5
step	0.27	0.3 / 0.5
facilitation		
max_fac	0.5	0.5 / 0.5
norm	0	0
tau [ms]	50	0.3 / 0.5
step	0.27	0.3 / 0.5

weights	mean [nS]	spread
recurrence		
$e \rightarrow e$	0.35	0.6 / 0.7
$i \rightarrow i$	1.00	0.6 / 0.7
$e \rightarrow i$	0.25	0.6 / 0.7
$i \rightarrow e$	1.50	0.6 / 0.7
input		
excitatory	6.0	0.6 / 0.7
inhibitory	20.0	0.6 / 0.7

Table VI.2: *Short term plasticity parameters (left) and weights (right) used for the software simulations. The spread is given relative to the corresponding mean in the form* relative standard deviation / relative bound*. If the value, drawn from a normal distribution, exceeds the bounds, it is redrawn form a uniform distribution within the bounds. The given weights reflect the base values for static synapses that are multiplied with w_{rec} and w_{inp}, respectively. The conversion to the weight of dynamic synapses is performed as described in section IV.5 with the reference rate chosen to be $\nu = 50\,\mathrm{Hz}$.*

Numerous *parameter sets*, each consisting of V_{rest}, w_{inp}, w_{rec}, c_{scale} and `syns`[13], reflecting different probability distributions and connection probabilities, were investigated. For each parameter set, 25 different recurrent networks (each with a different stimulus) were simulated yielding an overall simulation time of almost 2.3 million biological seconds. The simulation was performed on the cluster of the TU Graz.

VI.3.3 Evaluation of the PCSIM-Simulation

In the following, the results of a subset of the 9152 simulated parameter sets are presented. In order to quantify the network correlation, a measure was developed by the author with assistance of Michael Pfeiffer.[14]

A Measure of Network Correlation

The intent of the following correlation measure lies in quantifying simultaneity of the spike activity of a set of neurons. Its development was guided by the idea, that as little arbitrary constants as possible should be used. *Discretizations* for numerical calculations (e.g. bin sizes, time steps, ...) should not impose any meaning on the result, which needs to converge to the exact value when increasing numeric precision. The regarded spike

[13] whether *dynamic* or *static synapses* were used for recurrent connections.

[14] pfeiffer@igi.tu-graz.ac.at

trains should not be restricted to equal size or firing rate. Furthermore, the measure should reflect both *collective activity* and *inactivity*. Finally, it is assumed that if a number of spike trains, resulting from a certain set of conditions and yielding a correlation c, is expanded to even more spike trains, generated from the same condition set, a measure of correlation should converge towards a finite value, reflecting the inherent correlation caused by the underlying conditions. As stated above, the following suggestion has not yet been elaborated completely.

First, all spike trains are pre-processed separately, following a concept used in [*Maass et al.*, 2002] for describing network states. An exponentially decaying linear filter is applied to the spike trains, i.e. each spike adds 1 and decays exponentially in time reflecting the trace of the spike in the network. A reasonable decay time constant τ is the membrane time constant τ_{mem} or the synaptic time constant τ_{fall}, depending on which effect lasts longer. **Then**, each filtered spike train is normalized such that (a) its mean value equals zero (by subtracting $\tau \cdot$ #spikes/time from the filtered trace) and (b) its standard deviation over all time points equals one (by dividing the trace by its standard deviation[15]). **Finally**, for each time point the standard deviation is calculated over the set of such prepared spike trains. 'One minus the mean value of these standard deviations' is taken as the *correlation* of the set of spike trains.

In the following, some examples point out the basic properties of this measure.

Figure VI.3 shows three Poisson spike trains with rates 10 Hz, 20 Hz and 50 Hz and the corresponding normalized and filtered traces. This set of spike trains yields a correlation $c = 0.247$. 1000 Poisson spike trains with a mean rate of 20 Hz lasting for 1 second exhibited $c = 0.0097$. Generally, the correlation of randomly generated spike trains tends to zero for large samples (i.e. long duration, high firing rate, many neurons). Any spike train's autocorrelation is one.
Figure VI.4 shows four spike trains, each emitting a spike every 200 ms. In the upper case, the spike trains are clearly shifted yielding a correlation of $c = 0.145$. In the lower case, they are almost conjunct resulting in $c = 0.775$.
As the synaptic time constant exceeds the membrane time constant in the simulations, it is set $\tau = \tau_{fall} = 30$ ms for preprocessing the spike trains, in the following. Since this measure has not yet been systematically studied, the presented results should be accepted with caution.

Results

Figure VI.5 and figure VI.6 show the results of the software simulation. For clarity, only two network types $(w_{rec}, c_{scale}) = (0.5, 0.01)$ – parameter sets yielding almost input driven networks – and $(3.0, 0.03)$ – strong recurrence – are presented, as they offer all essential properties. The remaining configurations fit smoothly in this frame.

[15] If no spike occurred – being equivalent to a vanishing standard deviation – no normalization is necessary.

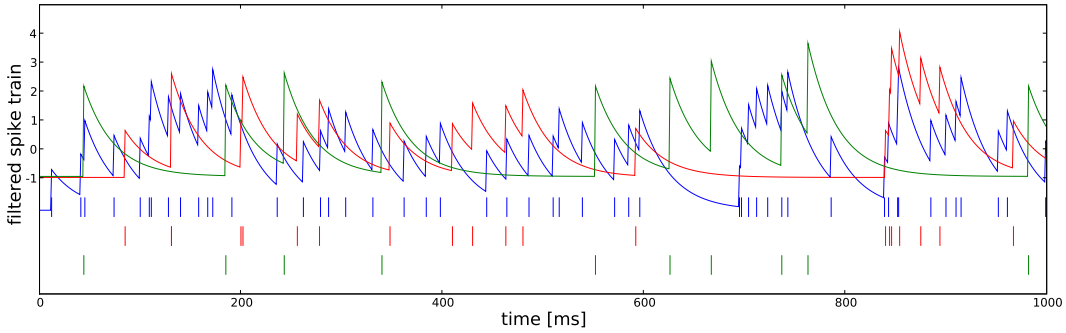


Figure VI.3: *Three Poisson spike trains with rates* 10 Hz, 20 Hz *and* 50 Hz *and their normalized and filtered traces* ($\tau = 30\,\mathrm{ms}$). *The correlation of this set of spike trains is* $c = 0.247$.

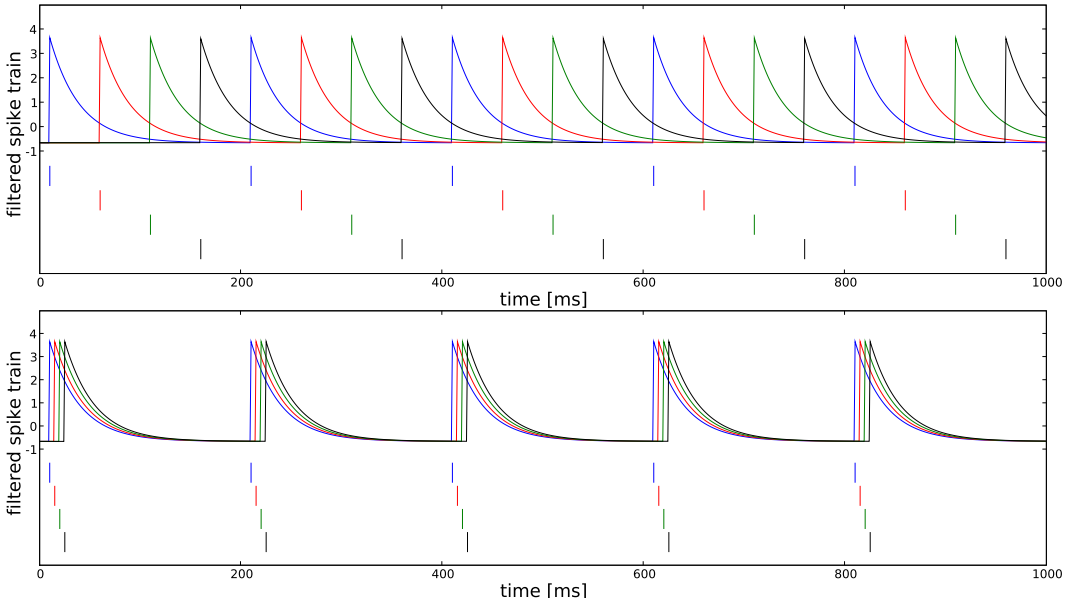


Figure VI.4: *Four spike trains firing equidistantly every* 200 ms. *For maximal distant spike trains (top) the correlation* $c = 0.145$ *is significantly lower than for almost conjunct spike trains (bottom) yielding* $c = 0.775$.

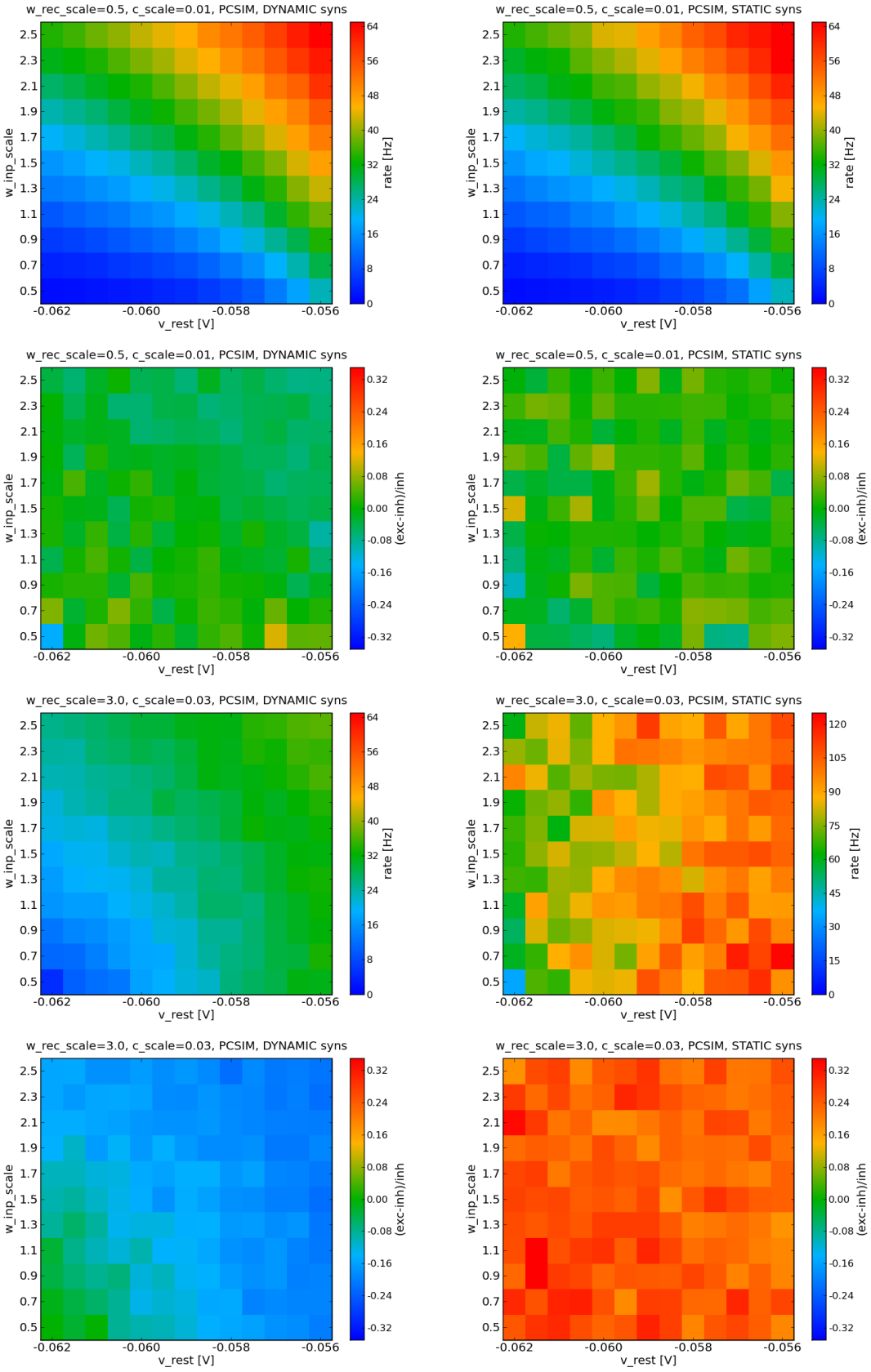
All figures follow the same design: The axes reflect different strengths of the stimuli: V_{rest} models the influence of the average stimulation, w_{inp} varies the strength of the fluctuations. The colors display one of the measures introduced above. Although the arrangement of the figures follows a clear pattern, each plot is additionally super-scribed with essential parameters used. Each pixel represents 25 simulated networks that had been generated from the same probability distributions and connection probabilities.

The first row of figure VI.5 proves that input driven networks exhibit no difference between dynamic and static synapses. Both a higher mean value and a larger variance of the external stimulation lead to an increased network activity.
This statement is confirmed by the population disparity $\frac{\nu_e - \nu_i}{\nu_i}$[16] shown in the second row.
The third row shows the overall mean activity for recurrent networks, connected with dynamic synapses (left), and static synapses (right). While the latter easily diverge to permanent firing – *mind the color bar ranges!* – the recurrent networks tune their activity to about 30 Hz: For weak stimuli activity is raised, for strong stimuli it is moderated, compared to input driven networks.
This property of dynamic synapses is reflected in the population disparity (fourth row). The stronger the input is, the more the networks respond with inhibitory activity. In contrast, networks with static synapses ignore changes in the input. Since after every spike, a neuron is reset to $-80\,\mathrm{mV}$ the maximum firing rate is limited.
These results prove network architectures recurrently connected with dynamic synapses to be capable of self-tuning their activity.

The second intent of 'self-stabilization' is addressed in figure VI.6. For each 25 simulated networks of a parameter set, the average activity was determined separately. The upper row shows the standard deviation of these 25 network activities (normalized to the mean of the parameter set). This inter-network variation is a measure of the steadiness or predictability of a network architecture. Obviously, dynamic synapses allow for generating networks with rather predictable properties. Static synapses exhibit significantly larger fluctuation from network to network.
The lower row of figure VI.6 shows the mean of the correlation within each of the 25 networks of a parameter set. In terms of the above described measure, networks based on dynamic synapses yield a lower synchronization than those based on static synapses, but still exceed the correlation of input driven networks (not shown).

In summary, the presented network architecture fulfills the basic requirements on a reliable and balancing 'substrate on the network level' even for rather small networks.

[16] Reading: 'The excitatory population fires at X percent higher / lower frequency than the inhibitory population.'



88

Figure VI.5: *Results of the software simulation. Comparison between* dynamic *(left) and* static *(right) synapses. For* almost input driven networks *(upper half) and* recurrent networks *(lower half) the* overall activity *(odd rows) and the* population disparity *(even rows) are shown.*

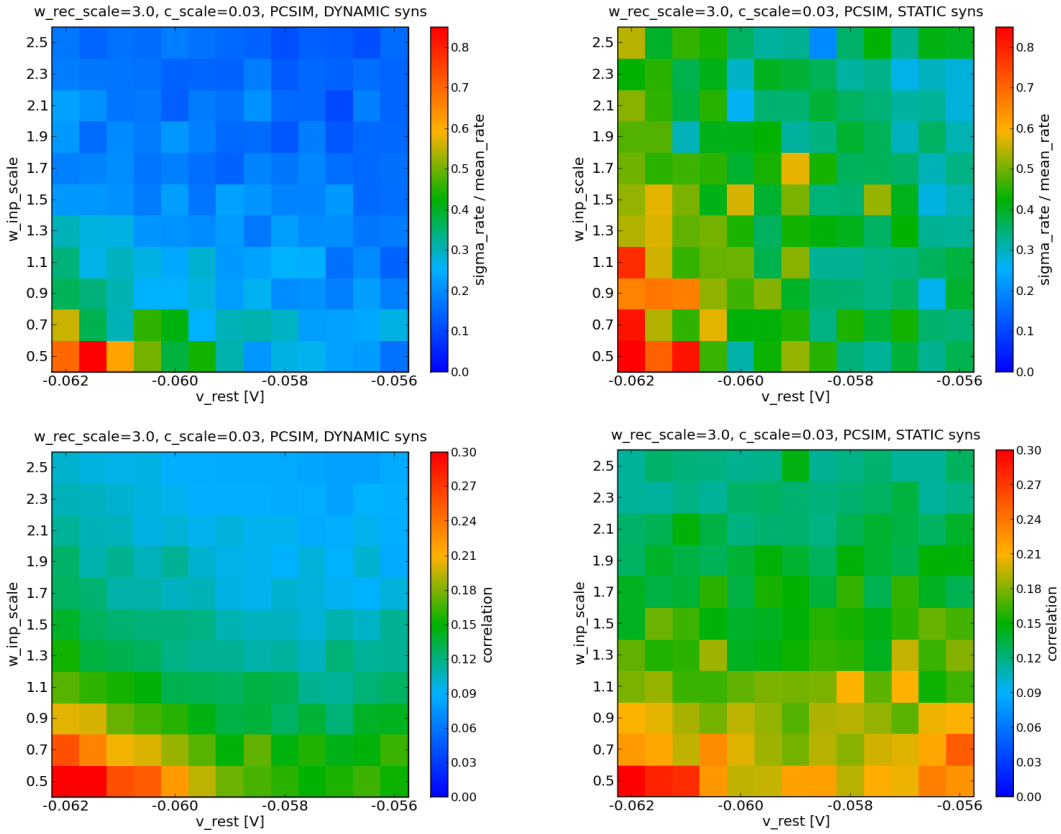


Figure VI.6: *Results of the software simulation. Comparison between* dynamic *(left) and* static *(right) synapses. The* inter-network variation in units of the mean firing rate of the respective parameter set *(top) and the* mean network correlation *(bottom) are shown. The regarded parameter sets correspond to those shown in the lower half of figure VI.5.*

VI.3.4 Transfer to the Hardware

As the above simulation results prove that even small generically generated recurrent networks can reliably stabilize their activity, the concept was transcribed to the FACETS Stage 1 Hardware. All scripts are written in the common meta language *PyNN* (see section II.1) except for the configuration of short term plasticity, since no low-level PyNN-interface has yet been defined. Thus, the preliminary 'PyNNesque' implementation presented in section V.2 was used.

As stated in section I.4, *Spikey III* suffers from an erroneous spike readout, limiting the number of *arbitrary* simultaneously recordable neurons to *three* for the considered network topology. But since every hardware neuron reveals an individual behavior, it is indispensable to record subsets randomly drawn throughout the entire network in order to avoid systematic errors when determining the average network activity. Therefore,

every network was run repeatedly with 20 randomly chosen sets of recorded neurons. Additionally, each such setup (network + recorded neurons) was run 5 times to regard analog fluctuations. This method is rather efficient on the hardware, since no reconfiguration must be applied to the chip. Hence, only the emitted spikes need to be read back reducing the serious configuration overhead.[17]
Due to these issues, 100 hardware runs are needed to acquire the average firing rate of a single network. Therefore, the number of networks generated for each *parameter set* was reduced to 20 and the density of the scanned parameter grid was diminished. Furthermore, only firing rates are considered for the evaluation of network properties, as spike trains recorded in different runs do not allow for measurements of the network correlation. In total, more than 2.8 million biological seconds were emulated on Spikey III no. 25.

Adjustments Influencing the Network Dynamics

If not stated otherwise, the configuration used in the computer simulation was maintained for the hardware emulation. But some parameters were changed for different reasons. A fundamental change had to be applied to the short term plasticity mechanism: As found in section IV.5, restrictions on the range of the `vout` voltages impede a precise adjustment of the STP circuitry. While the presented workarounds allow for a decent operation, control and knowledge over some parameters, e.g. V_{fac}, are lost. Hence, all STP-parameters needed to be re-adjusted. Particularly, it was impossible to perform an analytical conversion between the weights of static and dynamic synapses. Therefore, the correct weight transformation factors were measured directly on the oscilloscope for the applied STP configuration.[18] Measured membrane voltage traces, showing the used synapse dynamics, are presented in figure I.5. The applied setup is found in the caption of this figure.
Furthermore, as the experimental synapse driver calibration algorithm introduced in section IV.2 was used, the discrete hardware weights have not yet been mapped to their biological counterparts. Accordingly, new base values for all synaptic weights had to be set. An estimation of the synaptic efficacies suggests that the applied values – listed in table VI.3 – clearly differ from the weights used in the computer simulation.[19]

Finally, the external stimulation was slightly modified, as it was found to yield only low firing rates in case of purely input driven networks: The average rate of the input channels was increased to 12 Hz. The total number to inputs was raised to 64 and

[17] The actual emulation of 5 biological seconds takes only 50 μs with a `speedup` of 10^5. The upcoming *Gigabit Ethernet connection* will drastically lower the time needed for configuration and readout allowing for an immense acceleration of the presented experiments.
[18] Reference rate $\nu = 30$ Hz, measured multipliers for dynamic synapses: 1.125 (fac) and 1.375 (dep).
[19] Anyhow, the used hardware configuration is not the result of extensive parameter tuning. Essential improvements, used in the emulation, counteract the instability of the `VREST` voltage (see section IV.1) discovered in November 2008. Hence, no deep investigation of the hardware behavior was possible. Thus, the results presented below give rise to the assumption that a wide range of network configurations yields self-stabilizing properties.

weights	mean	spread
recurrence		
$e \to e$	3	0.6 / 0.7
$i \to i$	2	0.6 / 0.7
$e \to i$	1	0.6 / 0.7
$i \to e$	1	0.6 / 0.7
input		
excitatory	1	0.6 / 0.7
inhibitory	1	0.6 / 0.7

Table VI.3: *Weights used for the hardware emulation in units of discrete hardware weights w. The spread is given relative to the corresponding mean in the form* relative standard deviation / relative bound*. If the value, drawn from a normal distribution, exceeds the bounds, it is redrawn form a uniform distribution within the bounds. The given weights reflect the base values for static synapses that are multiplied with w_{rec} and w_{inp}, respectively. The conversion factors to the weight of dynamic synapses were measured directly on the oscilloscope. For $V_{mem} \approx -60\,\mathrm{mV}$, excitatory and inhibitory synapses yield similar EPSP- / IPSP-integrals (absolute values) when the same hardware weight is used.*

the fraction of excitatory channels to 75%. Each neuron received input from $3-6$ excitatory and $3-6$ inhibitory spike sources. Hence, the concept of equipollent stimuli was maintained.

VI.3.5 Evaluation of the Hardware-Emulation

Figure VI.7 exemplary shows the spike trains of a subset of 30 neurons of a randomly chosen trial of the hardware emulation. Only neurons emitting at least one spike are shown. The red line separates inhibitory (above the line) and excitatory neurons (below). While in the network with static synapses many neurons fire at maximum, dynamic synapses yield a rather sparse pattern. It is noticeable that only few neurons feature a 'Poisson-like' behavior, but neurons tend to emit bursts lasting for about 100 ms. Similar patterns were also observed in the software simulation and do not describe a hardware characteristic. As explained above, no systematic measurement of the network correlation was possible on the hardware. But manually observed spike pattern suggest that the emulations do not exhibit serious synchronization.

Figure VI.8 shows the results of the hardware emulation. As per the presentation of the software simulation results, just a subset of the parameter sets is shown. For all measures applied, only spikes emitted in the period $t \in [1\,\mathrm{s}, 4.5\,\mathrm{s}]$ were considered. The arrangement of the figures follows the above used pattern.
The first row shows the average activity of purely input driven networks. The configu-

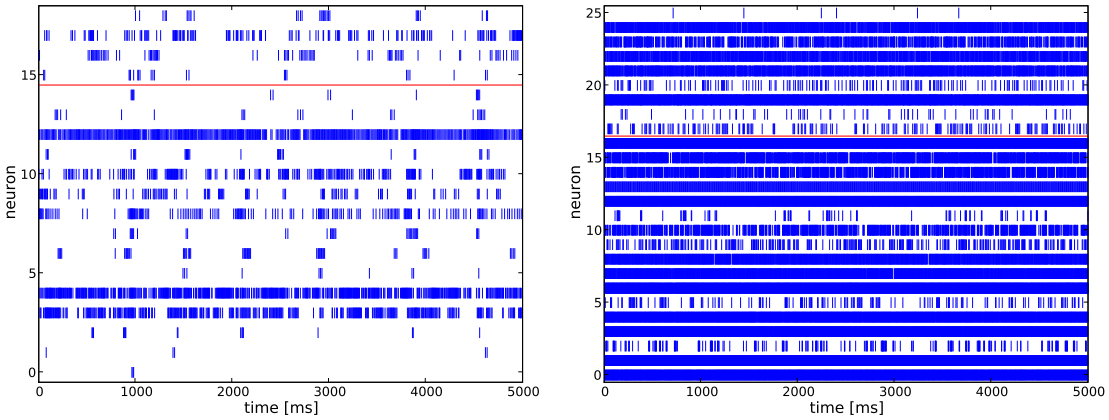


Figure VI.7: *Exemplary spike trains of recurrent networks with* dynamic *(left) and* static *(right) synapses emulated on the hardware. 30 randomly drawn neurons were recorded, but only those actually spiking are shown. The red line separates the inhibitory from the excitatory population. Both networks were generated from a parameter set, also considered for other evaluations:* $V_{rest} = -57\,\mathrm{mV}$, $w_{inp} = 7.0$, $w_{rec} = 2.5$ *and* $c_{scale} = 0.03$.

ration of the recurrent synapse driver dynamics has no influence on the network when all weights are cleared.[20]

More interestingly, the population disparity shown in the second row reveals a chip inherent discrepancy between the neurons assigned to the excitatory and the inhibitory population. As both populations are stimulated similarly, no systematic preference of excitatory activity should occur. Since for the applied external weights, the network emits almost no spike over a wide range of stimulation strength, the population disparity has not been normalized. Instead, the difference between both population's activities is shown.

The third row shows the mean firing rate of networks with rather strong recurrent connections. Particularly, the connection probabilities equal those of the referred computer simulated networks. Again, different ranges were used for the color bars of dynamic and static synapses. Clearly, short term plasticity allows for self-tuning of the networks towards about 15 Hz, while static synapses lead to a high activity.

When the recurrent networks raise activity, compared to the input driven setup, a higher-than-average excitatory rate is observed in case of dynamic synapses, as shown in the fourth row. This disparity continuously changes towards an increased inhibitory firing rate, as the external stimulation gets stronger. Remarkably, the networks are capable of compensating for the chip inherent preference of excitatory activity. In contrast, the vast majority of spikes arises from excitatory neurons, when using static synapses.

[20] While sounding trivial, such facts should be checked at the current state of development. Many inaccuracies have been discovered under similar conditions, when a specific network configuration was applied.

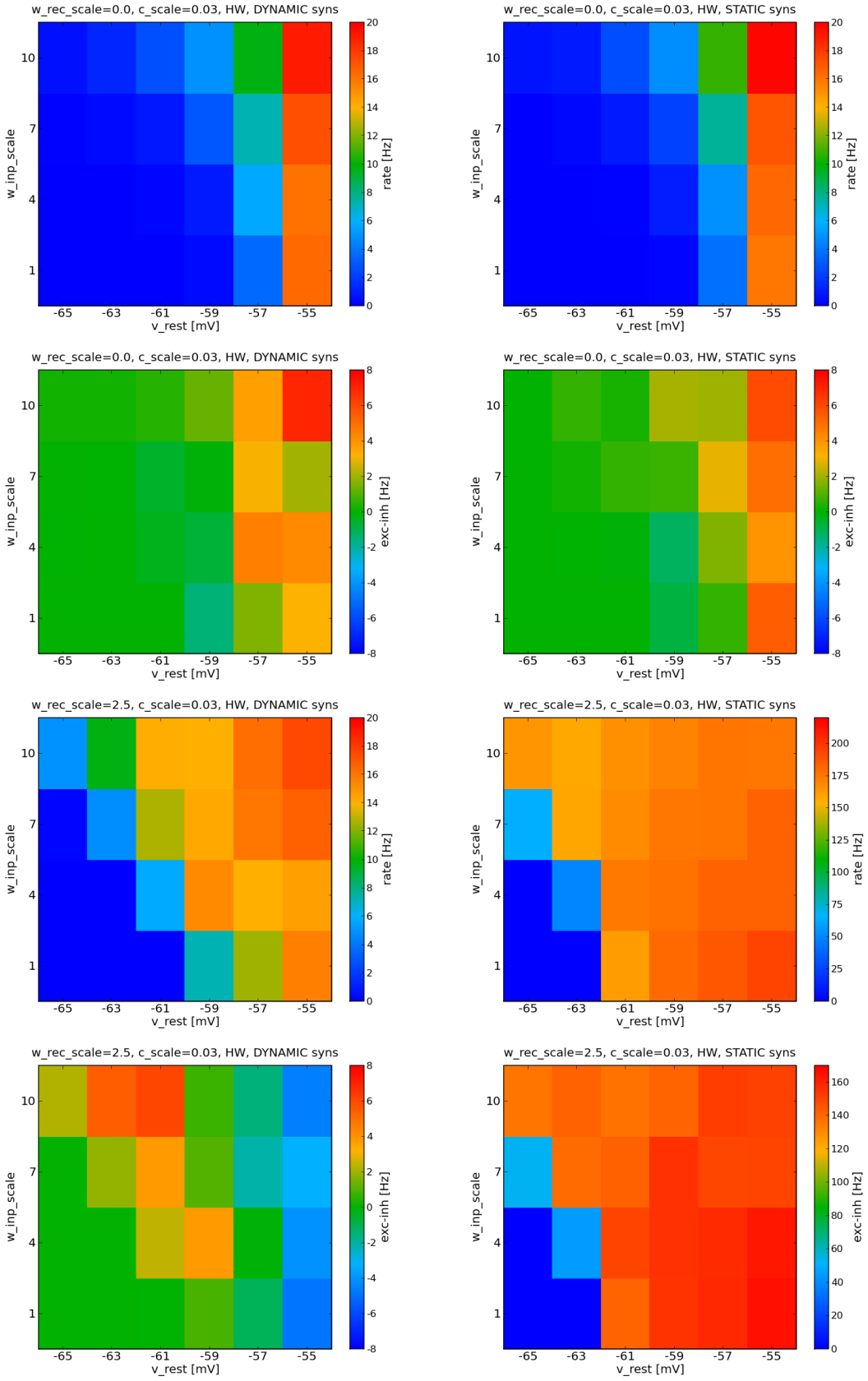


Figure VI.8: *Results of the hardware emulations. Comparison between* dynamic *(left) and* static *(right) synapses. For* purely input driven networks *(upper half) and* recurrent networks *(lower half) the* overall activity *(odd rows) and the* population disparity *(even rows) are shown.*

As the 'self-tuning property' of the averaged values is proven to be fulfilled, figure VI.9 compares the similarity of networks generated from the same probability distributions for the software simulation (left) and the hardware emulation (right). Only a comparable subset of parameter sets is considered. Especially, input driven networks are sorted out.

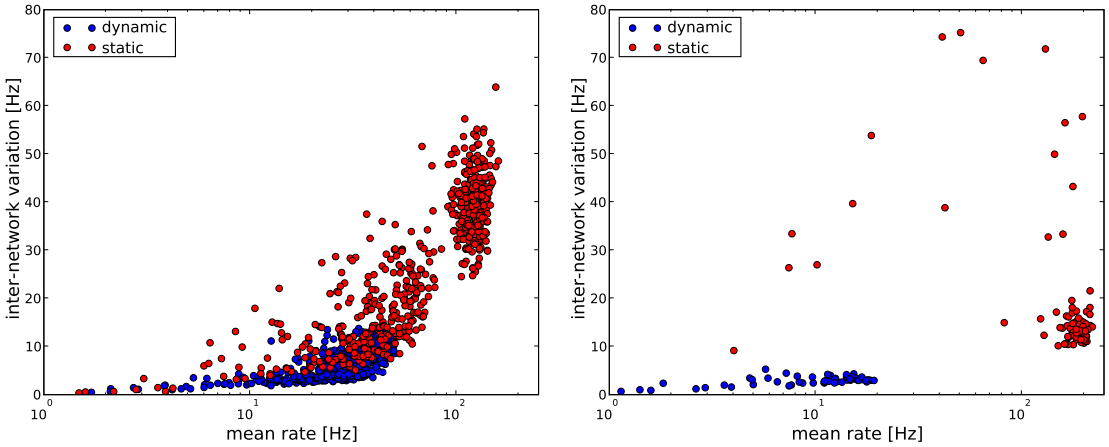


Figure VI.9: *The* inter-network variation *plotted versus the* mean activity*. Each dot corresponds to one parameter set. Only parameter sets, yielding a mean firing rate of more than* 1 Hz*, are shown. Furthermore, only a comparable subset of network configurations is considered.* Software (left): *All networks with* $w_{rec} \in \{2.0, 4.0\}$ *and* $c_{scale} \in \{0.02, 0.03\}$ *are used.* Hardware (right): *All networks with* $w_{rec} \in \{1.25, 2.5\}$ *and* $c_{scale} \in \{0.02, 0.03\}$ *are used.*

For each parameter set (one dot in the figure), the standard deviation of the mean firing rates of all 20 emulated networks[21] is plotted versus the overall mean activity of the respective parameter set. Particularly, for dynamic synapses the similarity regarding the steadiness of the network architecture is remarkable. In the regime of moderate activity, not a single configuration template features a predictable network behavior, when applying static synapses – the fluctuations clearly exceed the average firing rates.

Hence, the presented network architecture, using dynamic recurrent synapses, provides a 'substrate on the network level' for reliably counterbalancing inherent variations of hardware components. The practicability of the concept was prepared by extensive computer simulations and verified on the hardware. Further investigations are needed to reveal whether this approach can be employed for other network architectures without interfering with their specific properties. Possibly, the application of short term plasticity can serve as a foundation when exploring computational paradigms on the FACETS Hardware.

[21] 25 networks per set, in case of the software simulation.

VI.3.6 Further Tasks

To expand the investigation of the above described network architectures' properties, further studies – regarding single network dynamics as well as characteristics of the underlying probability distributions – need to be carried out.

For instance, the same parameter sets should be applied to different Spikey chips, in order to fathom the universality of the concept. During all simulations performed, the relative synaptic weights of the different connection types were fixed. Knowledge of the parameter ranges yielding a stable operation is essential, when adopting this approach to other network architectures.

Furthermore, the dynamics of individual networks should be examined, for determining their applicability. Thus, the variation of firing rates over time and the response to temporal changes in the input are crucial network properties. Will a network's average behavior be affected, if single synapses or neurons are reconfigured? And how do permutations in the mapping process influence the emulation results on the hardware?

Finally, the (auto-)correlation between consecutive hardware emulations of the same network stimulated with identical input touches the reproducibility of experiments. In case of a high (auto-)correlation, the measurement of the overall network correlation from data, acquired from different runs, would be possible.

Some of the functions, being needed to analyze the above issues, have already been implemented by the author, and are likely to be applied in the near future.

Conclusion

As discussed at the outset, every approach of neural network simulation features different benefits and disadvantages. For instance, the precision of the implemented neuron model either limits the network size or requires high computational efforts. The emulation of large networks on a highly accelerated analog hardware substrate takes a position at one extreme of the spectrum of available back-ends. The integration of the FACETS Hardware into the set of established simulation back-ends, will primarily depend on the successful reduction of inaccurate emulation results.

On this account, various types of inhomogeneity on the *Spikey III* chip were investigated.

A major issue addressed was an interference of the synapse drivers. Depending on their configuration, a chip-wide synaptic voltage was affected. Caused by the instability of this voltage, an erroneous permanent synaptic leakage conductance seriously shifted the neurons' resting potentials. Thus, the usage of one unit disturbed the dynamics of other units. Temporarily, this problem can be dealt with by limiting the range of the synaptic time constants. Furthermore, a workaround was presented that could minimize this error.
In order to reduce the dependency of network behavior on permutations in the mapping process, a well-performing routine aiming at the equalization of synaptic efficacy was developed. Additionally, the used concept allowed an approximation of the maximal performance calibration algorithms could possibly achieve. Variations within the synapse array define a lower boundary for the synaptic homogeneity the emulator can provide.
Similarly, variations of neuron parameters were discovered. Namely, the threshold voltage and the reset mechanism differed from neuron to neuron. In combination with the erroneous synaptic leakage conductance, the effective resting potential of some neurons exceeded the effective threshold voltage resulting in permanent firing. A calibration concept, introduced by the author and upgraded by Daniel Bruederle, allows to reduce these variations. Since the threshold comparator and the reset mechanism are correlated, such that a positive adjustment of one results in a worsening of the other, the possible performance of any calibration approach is limited.
Finally, Spikey's temporal homogeneity was investigated – more precisely, the correlation between temperature and network behavior. It was found that the chip configuration has no significant influence on its temperature. This is in contrast to the ambiance, which highly affects chip temperature and causes serious fluctuations in the network activity of some chips. Long term investigations yielded an increased activity during the night, which is likely to be caused by a drop of temperature. Multiple approaches, aiming at the stabilization of the chip temperature, were presented, none of which delivers a

satisfactory performance.

The reduction of such variations to a moderate extent is necessary for a reliable operation of the hardware.

The short term plasticity mechanism of the FACETS Stage 1 Hardware was included in the configuration flow from the low-level software to the PyNN-API. The dynamics of facilitating and depressing synapses were analyzed theoretically in order to gain control over the applied configuration. As it was found that a decent operation of dynamic synapses is hindered by a too narrow range of achievable programmable voltages, two workarounds were utilized to provide appropriate dynamics.

Following the credo of the PyNN-project to provide portable experiment descriptions, the `pyNN.hardware.stage1` module was improved and maintained. For the same purpose, hardware specific synapse models were added to the PCSIM simulation back-end.

Finally, two fundamentally different network architectures were emulated on the FACETS Stage 1 Hardware. Synfire chains exhibit a rather high tolerance to variations in synapse properties, allowing the emulation of closed-loop chains of remarkable stability. Since Spikey's topology does not offer the connectivity for multiple interconnected chains, the synchronization of synfire chains could not be studied.

By employing a generic self-stabilizing recurrent network architecture, it was investigated whether the above efforts to reduce hardware inhomogeneities are sufficient to provide a substrate on the network level with reliable properties. As a necessary preparation, a concept for external stimulation, with respect to various hardware constraints, was developed, allowing the emulation of high-conductance states while keeping stimulus correlation low.
To prove the applicability of the architecture to small-sized networks, extensive computer simulations were carried out during a research visit at the FACETS member TU Graz. As the concept was found to be feasible, it was applied to the chip. Like the PCSIM simulation, the hardware emulation featured two different aspects of self-stabilization of recurrent architectures: On average, the networks adjusted their activity to a certain level. Furthermore, the individual networks, generated from probability distributions, exhibited only small differences in their mean firing rates.
For both the PCSIM simulation and the hardware emulation, the performance of static and dynamic synapses was compared. While on both back-ends networks using static recurrent synapses were subject to large fluctuations, the dynamically connected architecture featured the above self-stabilization properties in either case. Thus, network substrates that are recurrently connected with dynamic synapses are likely to be able to reliably counterbalance inherent variations of hardware components.

In the end, the efforts which were invested in the reduction of unintended inhomogeneities of the Spikey chip were rewarded with success. However, finding these problems – and

understanding them – required once again a descent to technical details.
It remains a subject for further exploration, to what extent the above architectures are able to serve as a universal network substrate. For the presented first investigation, all possible requirements to a network were subordinated to the property of self-stabilization. But most concepts, which provide a high computational power or feature other characteristics, come along with specific requirements and constraints. Therefore, a universal substrate on the network level needs to be adaptable to a wide range of parameters and connection densities. Obviously, the lower the necessity to counterbalance back-end specific inhomogeneities is, the more universal any balancing network architecture becomes.
Moderating chip inherent variations of Spikey III required not only the application of all available calibration algorithms and several workarounds, but also the avoidance of critical hardware configurations. Thus, it is unlikely that the current revision of the FACETS Stage 1 Hardware is able to serve as a common back-end for the emulation of neural networks.

Being aware of this issue, it will be essential to both understand the causes of the current instabilities *and* to recheck the success of the applied improvements thoroughly. Eventually, the path from the discovery of erroneous network behavior to the comprehension of interactions on a highly technical level turns out to be both demanding and inefficient. However, the efforts undertaken during the last months, including the specification of numerous imperfections and the search for possible solutions, can pave the way towards an appreciable improvement of Spikey IV.

Outlook

Assuming that Spikey IV will exhibit a higher degree of homogeneity as well as a wider range of parameter adjustability than the current system, it can be assumed that a rather precise emulation of neural networks will be possible. But in practice, the FACETS Stage 1 Hardware currently lacks another issue that was supposed to be one of its major advantages: speed. The high acceleration of the emulation is compensated by two different problems. First, the communication to the hardware – in particular, sending and reading of spike trains – is much too slow and takes most of the runtime. Presumably, this problem will be solved soon via an Ethernet connection. Secondly, as not all neurons can be recorded at the same time, the identical setup has to be run repeatedly to gain sufficient information of the network state. But even if this fault can be fixed, the limited bandwidth of the Nathan bus requires a reduction of the hardware `speedup` from 10^5 to approximately 10^4. To slow down the neuron dynamics it is suggested to increase their membrane capacitance by the respective factor.

So, what tasks could a fast and rather homogeneous Spikey IV possibly solve better than other simulation back-ends? Relating to a suggestion of Prof. Dr. Wolfgang Maass, the computational power of sparsely connected recurrent networks could be investigated systematically. It is supposed that such architectures are able to perform computations of almost any degree of complexity. An yet unanswered question is *how* recurrent networks should be set up to yield optimal performance. One auspicious approach could be the comparison of 'better' and 'worse' networks in terms of statistical measures regarding, e.g. connection probabilities. Therefore, many different networks need to be generated from specified probability distributions and trained to several tasks. By an iterative procedure, well-performing construction principles can be dissected.
Since successful experiments have been carried out with rather small networks, consisting of about 100 neurons, Spikey provides a sufficiently large substrate. Furthermore, it is stated that it is diversity that makes this approach powerful. So, a limited amount of inhomogeneity of the substrate, can be assumed to not interfere with the results. Therefore, this could be an application like tailored for the FACETS Hardware.

Looking a bit further into the future, the same concepts should be applied to much larger networks using the upcoming Stage 2 wafer-scale integration system. This device is designed to allow the emulation of entire cortical domains, like the primary visual cortex V1. Furthermore, the spike-time dependent plasticity mechanism renders long-term learning experiments possible, utilizing the full acceleration of the hardware.
Finally, due to the reliable and constant speed of generated output accompanied by a low power consumption, analog hardware emulators might find a technical application

as embedded systems. Furthermore, such devices might provide a chance to exploit the computational potential of upcoming, very small structures of electronic units – which are prone to high production error rates – since network architectures are able to self-adjust to imperfections of the substrate.

Bibliography

Abeles, M., *Corticonics: Neural Circuits of the Cerebral Cortex*, Cambridge University Press, New York, 1991.

Baddeley, R., L. F. Abbott, M. C. A. Booth, F. Sengpiel, T. Freeman, E. A. Wakeman, and E. T. Rolls, Responses of neurons in primary and inferior temporal visual cortices to natural scenes, *Proceedings of the Royal Society B*, *264*, 1775–1783, 1997.

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., A model of neocortex, *Network: Computation in Neural Systems*, *6*, 179–224, 1995.

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.

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., 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.

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

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*, 2008, accepted for publication.

Destexhe, A., D. Contreras, and M. Steriade, Mechanisms underlying the synchronizing action of corticothalamic feedback through inhibition of thalamic relay cells, *Journal of Neurophysiology*, *79*, 999–1016, 1998.

Destexhe, A., M. Rudolph, J. M. Fellous, and T. J. Sejnowski, Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons., *Neuroscience*, *107*, 13–24, 2001.

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., M.-O. Gewaltig, and A. Aertsen, 1999), title = Stable propagation of synchronous spiking in cortical neural networks, journal = Nature, volume = 402, pages = 529-533,.

EPFL, and IBM, Blue brain project, 2008.

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

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.

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

Gewaltig, M. O., M. Diesmann, and A. Aertsen, Propagation of cortical synfire activity: survival probability in single trials and stability in the mean., *Neural Netw*, *14*, 657–673, 2001.

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

Haß, J., S. Blaschke, T. Rammsayer, and J. M. Herrmann, A neurocomputational model for optimal temporal processing., *Journal of computational neuroscience*, 2008.

Hines, M., and N. Carnevale, *The NEURON simulation environment.*, pp. 769–773, M.A. Arbib, 2003.

Hines, M., J. W. Moore, and T. Carnevale, Neuron, 2008.

Hô, N., and A. Destexhe, Synaptic background activity enhances the responsiveness of neocortical pyramidal neurons., *J Neurophysiol*, *84*, 1488–1496, 2000.

Hodgkin, A. L., and A. F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve., *J Physiol*, *117*, 500–544, 1952.

Hubel, D., and T. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, *J.Physiology*, *160*, 106–154, 1962.

Hubel, D., and T. Wiesel, Receptive fields and functional architecture in two non-striate visual areas (18 and 19) of the cat, *J. Neurophysiology*, *28*, 229–289, 1965.

Huettel, S. A., A. W. Song, and G. McCarthy, *Functional Magnetic Resonance Imaging*, Sinauer Associates, 2004.

Jaeger, H., The "echo state" approach to analysing and training recurrent neural networks, *Tech. Rep. GMD Report 148*, German National Research Center for Information Technology, 2001.

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

Lehmann, K., and S. Löwel, Age-dependent ocular dominance plasticity in adult mice, *PLoS ONE*, *3*, e3120, 2008.

Maass, W., and E. D. Sontag, Neural systems as nonlinear filters, *Neural Computation*, *12*, 2000, 2000.

Maass, W., T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, *Neural Computation*, *14*, 2531–2560, 2002.

Maass, W., T. Natschlager, and H. Markram, Fading memory and kernel properties of generic cortical microcircuit models, *J Physiol Paris*, *98*, 315–30, 2004, institute for Theoretical Computer Science, Technische Universitaet Graz, Austria. maass@igi.tugraz.at.

Maass, W., P. Joshi, and E. D. Sontag, Computational aspects of feedback in neural circuits, *PLoS Computational Biology*, *3*, e165+, 2007.

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.

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.

Mountcastle, V. B., An organizing principle for cerebral function: The unit module and the distributed system, in *Neuroscience, Fourth Study Program*, edited by F. O. Schmitt, pp. 21–42, MIT Press, Cambridge, MA, 1979.

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

Neural Ensemble, Website, `http://neuralensemble.org`, 2008.

Pecevski, D., and T. Natschläger, PCSIM website, `http://sourceforge.net/projects/pcsim`, 2008.

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.

Rosenblatt, F., The perceptron: a probabilistic model for information storage and organization in the brain, *Psychological Review*, *65*, 386–408, 1958.

Rudolph, M., and A. Destexhe, Tuning neocortical pyramidal neurons between integrators and coincidence detectors, *J Comput Neurosci*, *14*, 239–251, 2003.

Sakmann, B., and E. Neher (Eds.), *Single-channel recording*, Plenum press, 1995.

Schemmel, J., personal communication, 2008.

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.

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.

Shu, Y., A. Hasenstaub, M. Badoual, T. Bal, and D. A. McCormick, Barrages of synaptic activity control the gain and sensitivity of cortical neurons, *Journal of Neuroscience*, *23*, 10,388–10,401, 2003.

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

Steriade, M., *The Intact and Sliced Brain*, MIT Press, 2001.

Steriade, M., I. Timofeev, and F. Grenier, Natural waking and sleep states: A view from inside neocortical neurons, *J Neurophysiol*, *85*, 1969 – 1985, 2001.

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

The NEural Simulation Technology Initiative, NEST website, `http://www.nest-initiative.org`, 2008.

Thomson, A. M., 1997.

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.

Wendt, K., M. Ehrlich, and R. Schüffny, A graph theoretical approach for a multistep mapping software for the facets project, in *CEA'08: Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications*, pp. 189–194, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2008.

Acknowledgments

I want to express my gratitude to all who made this work possible, especially:

Prof. Dr. Karlheinz Meier

Prof. Dr. Wolfgang Maass

Devine D

Mythical M

Brillant B

Excellent E

the entire Graz-Connection

and my Parents, Friends and DoubleYouGee

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 5, 2008

.....
(signature)