Dissertation

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics

of the Ruperto-Carola University of Heidelberg, Germany

for the degree of
Doctor of Natural Sciences

presented by
Dipl.-Phys. Daniel Brüderle
born in Offenburg, Germany

Date of oral examination: July 8, 2009

Neuroscientific Modeling with a Mixed-Signal VLSI Hardware System

Referees: Prof. Dr. Karlheinz Meier
Prof. Dr. Alain Destexhe

Abstract

Neuroscientific Modeling with a Mixed-Signal VLSI Hardware System

Modeling networks of spiking neurons is a common scientific method that helps to understand how biological neural systems represent, process and store information. But the simulation of large-scale models on machines based on the Turing paradigm is subject to performance limitations, since it suffers from an intrinsic discrepancy to the massive parallelism of neural processing in the brain. Following an alternative approach, neuromorphic engineering implements the structure and function of biological neural systems in analog or analog-digital VLSI devices. Neuron and synapse circuits represent physical models that evolve in parallel and in continuous time. Therefore, neuromorphic systems can overcome limitations of pure software approaches in terms of speed and scalability. Recent developments aim at the realization of large-scale, massively accelerated and highly configurable neuromorphic architectures. This thesis presents a novel methodological framework that renders possible the beneficial utilization of such devices as neuroscientific modeling tools. In a comprehensive study, it describes, tests and characterizes an existing prototype in detail. It presents policies for the biological interpretation of the hardware output and techniques for the calibration of the chip. The thesis introduces a dedicated software framework that implements these methods and integrates the hardware interface into a simulator-independent modeling language, which is also supported by various established software simulators. This allows to port experiment descriptions between hardware and software simulators, to compare generated output data and consequently to verify the hardware model. The functionality of the translation methods, the calibration techniques and the verification framework are shown in various experiments both on the single cell and on the network level.

Neurowissenschafliches Modellieren mit einer Analog-Digitalen VLSI Hardware

Die Modellierung pulsgekoppelter neuronaler Netzwerke ist eine übliche wissenschaftliche Methode um die Kodierung, die Verarbeitung und die Speicherung von Information in biologischen neuronalen Systemen zu verstehen. Bei der Simulation großskaliger Modelle auf Computern, die nach dem Turingprinzip arbeiten, ergeben sich jedoch Performanzeinbußen aufgrund der Diskrepanz zur intrinsisch hochparallelen Verarbeitungsweise im Gehirn. Einen alternativen Ansatz dazu stellen neuromorphe Hardwaresysteme dar, die die Struktur und Funktion biologischer neuronaler Systeme in analoger oder gemischt analog-digitaler hochintegrierter Schaltungstechnik emulieren. Die Neuronen- und Synapsenschaltungen sind dabei physikalische Modelle, die sich parallel und zeitlich kontinuierlich entwickeln. Daher sind neuromorphe Systeme in der Lage, die Geschwindigkeits- und Skalierungsbeschränkungen reiner Softwarelösungen zu überwinden. Derzeit werden großskalige, massive beschleunigte und hoch konfigurierbare neuromorphe Architekturen entwickelt. Diese Doktorarbeit präsentiert ein neuartiges methodisches Konzept, das die Verwendung solcher Systeme für neurowissenschaftliches Modellieren ermöglicht. Ein bereits verfügbarer Prototyp wird detailliert beschrieben und umfassend getestet. Es werden Techniken eingeführt, die es erlauben, die Ausgabe der Hardware biologisch zu interpretieren und den Chip zu kalibrieren. Software, die im Rahmen dieser Arbeit vorgestellt wird, implementiert diese Methoden. Sie wurde in eine simulator-unabhängige Modellierungssprache eingebettet, die auch von etablierten Software-Simulatoren unterstützt wird. Dadurch können Experimentbeschreibungen zwischen Hardware und Software-Simulatoren ausgetauscht, erzeugte Ergebnisdaten verglichen und damit auch das Hardwaremodell verifiziert werden. Es werden Experimente an einzelnen Neuronen und ganzen Netzwerken präsentiert, die die Funktionalität der Übersetzungs- und Kalibrierungsmethoden sowie des Verifikationsprinzips bestätigen.

Contents

Introduction

Investigating the brain is a scientific effort that does not need an extensive motivation. Understanding principles of neural information processing is a human concern as fundamental as questions on cosmology or particle physics. In contrast to this, the manifold and diverse *methods* that are applied in the field of neuroscience have to be permanently put into question. So far, the available approaches are not sufficient to reveal final explanations for all remarkable capabilities inherent to neural systems. Consequently, the technological aspects of neuroscience are subject to ongoing revisions, improvements and innovations. Section 1.2 lists techniques that significantly contributed to the current state of knowledge about neural systems.

Especially due to the technical and technological difficulties in accessing and monitoring the living brain, *modeling* represents an important approach within the spectrum of neuroscientific efforts. A large community of researchers develops models of different neural systems and thereby extracts important insights. These models represent various anatomical fractions of their biological original and incorporate different levels of detail regarding the utilized constituents and the applied structural complexity. Section 1.2.2 provides a selection of recent publications from that field.

One branch of the modeling community uses computers to numerically calculate the emerging dynamics of their models. Driven by the rapid development of available and affordable computational power during the last decades, the importance of this field has steadily grown. But the utilization of computers based on the Turing paradigm (Turing, 1937), i.e. that perform sequential transitions between discrete internal states using a small number of processing units, renders a discrepancy to the massive parallelism of analog neural computation. This implies performance problems (see e.g. Morrison et al., 2005), as will be discussed in Section 1.2.2.

An alternative approach is represented by implementing the structure and function of biological neural systems in analog or mixed-signal[1] VLSI[2] technology, often referred to as *neuromorphic systems* engineering (Mead, 1990; Cauwenberghs, 1999). Building upon the pioneering work of Carver Mead (Mead and Mahowald, 1988; Mead, 1989), electronic engineers have developed neuromorphic devices since the 1980s. They realized remarkable biologically inspired hardware implementations of spike-based information processing systems, such as silicon retinas (Delbrück and Liu, 2004; Serrano-Gotarredona et al., 2006), self-tuning motor control units for robotics (Lewis et al., 2000), attractor memory devices (Vogelstein et al., 2007), self-organizing pattern classifiers (Häfliger, 2007) and many more. Some further examples will be introduced in Section 1.2.3, and the potential, the advantages and the disadvantages of neuromorphic devices will be discussed.

But although the manifold applications are promising, neuromorphic engineering still represents a rather exotic niche within the variety of research fields in neuroscience. So far,

[1] Chips that incorporate both digital and analog circuitry are called *mixed-signal* devices.

[2] Very-Large-Scale Integration, i.e. the integration of circuits comprising thousands or millions of transistors on one single chip.

the ways in which neuromorphic engineers and modeling neuroscientists can mutually benefit from each other are not symmetrically exploited. Chip designers transfer biological principles to their semiconductor substrates, e.g. the massively parallel operation of simple computing units and locally operating plasticity rules (Bi and Poo, 1997; Markram et al., 1997; Song and Abbott, 2001; Morrison et al., 2008). They reproduce and exploit the intrinsic fault tolerance and the self-optimization features of such architectures (see e.g. Sussillo et al., 2007; Bill, 2008, Section 6.2.2). Still, the focus of most neuromorphic engineering efforts is rather application specific, and the technological development has not yet brought up a neuromorphic device that is flexible and large enough to serve as a neuroscientific modeling tool.

This situation might possibly change in the following years. Within the FACETS research project, which will be introduced in Section 1.4, a novel type of hardware is currently developed (see Schemmel et al., 2006; Ehrlich et al., 2007; Schemmel et al., 2008 and Section 2.2). Devices of that type will combine a massive acceleration, large network sizes and a high configurability with the advantages inherent to analog neuromorphic devices such as power efficiency and a time-continuous operation (see Section 1.2.3). Following this strategy, neuromorphic engineering has the potential to step out of its niche and provide new and relevant input towards the understanding of cortical dynamics.

Main Questions Addressed In This Thesis

A prototype for such a novel neuromorphic device is already available. It will be introduced in detail in Section 2.1. Based on experimental work with devices of this type, the thesis at hand is an effort to answer the following fundamental questions:

- Can neuromorphic hardware devices of the investigated type serve as modeling tools for neuroscience and provide *new* insights into neural information processing?
- What are the requirements for the acceptance of such neuromorphic tools in the established modeling community?
- What are the qualities of neuromorphic devices that have to be exploited in order to provide a benefit for modelers?
- Which methods have to be applied in order to translate between the hardware domains and the biological model?
- What are the technical challenges and obstacles on the way towards neuroscientific modeling with neuromorphic hardware?

This dissertation suggests a novel set of methods for the realization of neuroscientifically relevant models with neuromorphic hardware. For this purpose, the available prototype hardware system is deployed in various case studies, thereby testing the proposed paradigms.

The utilized chip is not an optimal device for this purpose due to its prototypic nature. It represents an early and thus imperfect developmental stage, and its resources, e.g. the number of neurons and synapses per chip, are limited. Nonetheless, the presented realization of the suggested paradigms yields essential practical experience and solutions for the overcoming of emerging obstacles. The insights gained from this thesis are expected to be useful for the development and operation of future systems.

Technical and Experimental Work The utilization of the FACETS prototype hardware device implies specific, device-related goals:

- To specify the impact of constituent variations inherent to the employed chip technology.
- To provide calibration methods that deal with these variations.
- To document prototype-specific and design-related problems and, if possible, provide methods to handle them.
- To share technical experience and provide recommendations for the operation of the system.
- To document experiments that have been performed with this device both on the cell level and on the network level.

Dedicated chapters will address these device-specific issues.

Structure of this Thesis

The present thesis is structured as follows: After this introduction, Chapter 1 will provide background information and references that sketch the status of current research in neuroscience from the point of view of neuromorphic engineering. Methods and models are introduced that are relevant for this thesis. In Chapter 2, the investigated neuromorphic hardware system is described in detail. Chapter 3 outlines scenarios of useful device operation. Based upon these, it describes paradigms for the translation between the hardware domain and the biological model, including the important concept of embedding the system interface into the modeling language PyNN. Furthermore, it describes the full software layer stack that has been implemented to realize the proposed concepts and techniques. Chapter 4 describes the investigation of process-inherent and design-related imperfections of the used chip. In this context, the functionality evaluation of certain sub-modules of the device requires methods for the measurement of specific variables that are not directly accessible. A set of such techniques has been developed and is presented in the beginning of the chapter. After the technical problems have been identified, Chapter 5 introduces methods to overcome these obstacles and to establish biologically realistic activity regimes on the hardware. This includes the handling of chip malfunctions, the calibration of unavoidable constituent variations and the gauging of hardware dynamics with established software simulators. In order to evaluate the biological relevance of a neural network experiment performed in hardware, statistical descriptors of network dynamics are provided which allow for a comparison between output data generated by hardware and software back-ends. Hardware experiments on the single cell and on the network level are described in Chapter 6. Utilizing the previously described methods, the calibration techniques and the introduced software framework, the experiments represent a first proof of concept for the developed neuromorphic modeling framework. A conclusion and discussion of the presented work is given at the end of this thesis, followed by an outlook on ongoing and planned work as well as on estimated further developments in the field.

Naming Conventions

The hardware device in focus of this thesis implements a physical model of cortical neurons with respect to their information processing dynamics. This model is *not* based on the numerical solving of sets of differential equations with digital logic, which is usually referred to as a *simulation*, but it is composed of microelectronic circuits, i.e. real physical objects which imitate the electric dynamics of real neurons. Therefore, employing such a hardware model will be called an *emulation*.

One important feature of the employed hardware device is that its intrinsic time constants are in the order of 10^5 times shorter than the corresponding values in the biological original. Hence, the hardware time domain has to be translated into the biological one and vice versa in order to make use of the model. The *hardware time domain* will be abbreviated with `HTD` throughout this thesis and corresponds to the physical lab time. Unless otherwise expressly mentioned, the applied interpretation (see Section 3.1.5) defines that any emulated biological time is exactly a 10^5-fold of the period measured in `HTD`. The *biological time domain* will be abbreviated with `BTD`.

Analogously, the membrane potentials, capacitances, conductances and currents in hardware can be translated to their biological counterparts by a linear transformation, which will be described in Section 3.1.5. In order to avoid confusions regarding these two electrical domains, the abbreviations `HVD` and `BVD` will be used, standing for *hardware* and *biological voltage domain*, respectively, but accounting for all transformed electrical dimensions as well.

The hardware devices which have been utilized for the presented work were developed within the research project FACETS (see section 1.4) and serve as prototypes for a much larger system currently under development. Therefore, the prototype system is referred to as the *FACETS Stage 1 Hardware*, and the system currently built as the *FACETS Stage 2 Hardware*. In order to avoid these bulky names throughout the text, they will be abbreviated with `FHW-1` and `FHW-2`, respectively. For the `FHW-1` chip, three versions exist already, i.e. the design has gone through two revisions. The short forms `FHW-1.1`, `FHW-1.2` and `FHW-1.3` will indicate the utilized chip version. For every chip version, multiple systems exist, which under ideal conditions would behave identically, but which, due to production variations, do not. Hence, for some experimental data, e.g. for chip-to-chip comparisons, the individual index of the employed chip will be given, e.g. `FHW-1.3-No.25`.

Cooperations

This thesis incorporates data, methods and findings that have been acquired in collaborations with various people.

The Hardware System The utilized `FHW-1` system has been developed by Dr. Johannes Schemmel[3] and further members of the Electronic Vision(s) group at the Kirchhoff Institute for Physics in Heidelberg, Germany, under the coordination of Professor Dr. Karlheinz Meier[3]. Dr. Johannes Schemmel, Dr. Andreas Grübl[3], Dr. Stefan Philipp[3], Dan Husmann[3] and Sebastian Millner[3] actively supported the work presented in this thesis by providing and

[3]Professor Dr. Karlheinz Meier, Dr. Johannes Schemmel, Dr. Andreas Grübl, Dr. Stefan Philipp, Dan Husmann and Sebastian Millner are with the Kirchhoff Institute for Physics, University of Heidelberg, Germany.

servicing the full hardware framework presented in Section 2.1. The experiments presented in Section 6.1.1 have been performed in close collaboration with Dr. Andreas Grübl.

Supervised Work The author supervised and coordinated the committed work of Bernhard Kaplan[4], Eric Müller[4] and Johannes Bill[4] for their diploma theses. Bernhard Kaplan worked on the hardware-specific high-conductance state analysis method presented in Section 4.1.2. Eric Müller contributed to the development of the software framework presented in Section 3.2 and performed measurements of long-term plasticity characteristics in hardware synapses presented in Sections 4.1.3 and 6.1.5. Johannes Bill helped specifying design-related chip malfunctions, contributed methods to handle them and worked on the development of self-tuning network architectures described in Section 6.2.2.

Further Collaborations The experiment series presented in Section 6.2.1 have been developed and performed in close collaboration with Jens Kremkow[5].

The work described in Section 3.1.3 contributes to the PyNN project, which is coordinated and inspired by Andrew Davison[6] and Eilif Muller[7].

[4]Bernhard Kaplan, Eric Müller and Johannes Bill are with the Kirchhoff Institute for Physics, University of Heidelberg, Germany.

[5]Jens Kremkow is with the Institut de Neurosciences Cognitives de la Méditerranée, CNRS, Marseille, France, and with the Institute for Neurobiology and Biophysics, University of Freiburg, Germany.

[6]Andrew Davison is with the Unité de Neurosciences Intégratives et Computationelles, CNRS, Gif sur Yvette, France.

[7]Eilif Muller is with the Laboratory of Computational Neuroscience, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland.

1 Neuroscience and Neuromorphic Engineering

In the following chapter, the current status of research in neuroscience is sketched (Sections 1.1 and 1.2). Neural network modeling approaches are introduced in Section 1.2.2. Two branches of modeling, namely pure software simulations and physical hardware implementations, are compared in Section 1.2.3. The advantages and disadvantages of both strategies are discussed, and requirements for the establishment of neuromorphic hardware systems as neuroscientifically useful modeling tools are described in Section 1.2.4. A selection of established models of membrane and synapse dynamics as well as models of synaptic plasticity that are relevant for the understanding of the hardware model utilized throughout this thesis are provided in Section 1.3. Since the development of this hardware system and all presented work are embedded into the FACETS research collaboration, the project is outlined in Section 1.4.

1.1 The World in the Mind

Every biological organism interacts with its environment. As a part of this process, the acquisition and the use of information about the surrounding world is essential for the survival of every living system. Many forms of life have developed a variety of highly complex organs to optimize both the perception and the processing of information. Sense organs and nervous systems allow to acquire relevant data and generate reactions based upon this input.

The *central nervous system* of an organism performs the integration of sensory input, the control of organism-internal processes and the coordination of motor actions. It intrinsically reflects the world as it is perceived by the individual creature, or, in other words, it represents a model of the niche in which the innervated organism lives. In most organisms, this model is rather static, i.e. actions occur mainly by means of genetically coded and evolutionarily optimized reflexes. In some species, though, the nervous system has developed to a structure that is capable of dynamically adapting its intrinsic world representation to changes in the environment within one individual's lifetime – the organisms can learn. Highly developed forms of such systems are brains, which contain dense networks of intercommunicating nerve cells, so-called *neurons*. Based on the information that is currently perceived and on what has been perceived in the past, brains generate predictions of what is going to happen next and can therewith generate *reason*able behavior.

Mammalian brains incorporate a structure that has turned out to be evolutionarily very successful, the *neocortex* (or simply *cortex*). The cortex is a layered nervous tissue embracing older (in terms of evolutionary development) parts of the brain. It plays a key role in the processing of sensory information, in motor control tasks, in the formation of memory, in attention and in awareness. In humans, it also significantly contributes to phenomena like language and consciousness.

For centuries, scientists of different disciplines have been researching the functionality of the brain, and today they have a large amount of partly very detailed knowledge to their disposal. Still, a conclusive explanation of many of the remarkable brain capabilities is missing.

1.2 Modern Neuroscience – Methods and Models

One of the goals of scientifically studying nervous systems is to understand the mechanisms that underlie the capabilities of neural structures. This section gives a depiction of neuroscientific techniques and advancements that have shaped the modern picture of the brain. Modern methods to access information about the anatomy and the functions of neural tissue are described with a focus on investigating the cortex. Modeling approaches which build upon this information, but which also contribute to the understanding of brain-like information processing, are introduced.

1.2.1 Studying the Brain

Charting the morphology and circuitry of the brain and monitoring the processes going on inside of it is still a technological challenge today. The cat visual cortex, for example, incorporates approximately 50,000 neurons per 1 mm^3, and each of these neurons connects to approximately 6,000 other cells via so-called *synapses* (Beaulieu and Colonnier, 1985). For a review of basic neocortical neuron and synapse types and their functions see e.g. Douglas et al., 2004. Here, only a very coarse summary of the most basic facts of neural and synaptic structure and function is given.

Information Processing Constituents

Cortical neurons consist of a cell body (or *soma*) with a diameter typically ranging from about 10 µm to 50 µm, and of morphological branching structures called the *dendrites* and the *axon*. Via the dendrites the neuron receives input *from* other cells, while along the axon it carries its output *to* other cells. Information exchange along these wire-like cell extensions is performed by sending so-called *action potentials* or *spikes*. Every neuron exhibits a dynamically evolving electrical potential difference across its cell membrane. Once this *membrane voltage* exceeds a certain *threshold* value, a sharp voltage peak (the action potential) will be generated (or *fired*) by the soma. This *pre-synaptic* spike travels along the axon and arrives at synapses that connect to the dendrites of *post-synaptic* neurons. In these synapses[1], the release of neurotransmitters is triggered, which changes the properties of ion-channels incorporated in the post-synaptic cell membrane and therewith possibly its firing behavior.

[1] Only the case of chemical synapses is considered here. See e.g. Douglas et al., 2004 for a description of *electrical synapses*.

In Figure 1.1, a stained cortical neuron is depicted. It shows an exemplar of the so-called *pyramidal* cell types, which represent the majority of *excitatory* cells in the cortex. The term *excitatory* describes the kind of impact that such a cell has on other neurons in case it sends an action potential: The described release of neuro-transmitters will cause a change in conductance and consequently in a current flow via the post-synaptic cell membrane that temporarily increases this post-synaptic cell's own firing probability. The opposite effect, i.e. the firing probability decrease of a stimulated neuron caused by pulling its membrane potential away from its firing threshold, is referred to as *inhibition*. Both the dendritic and the axonal

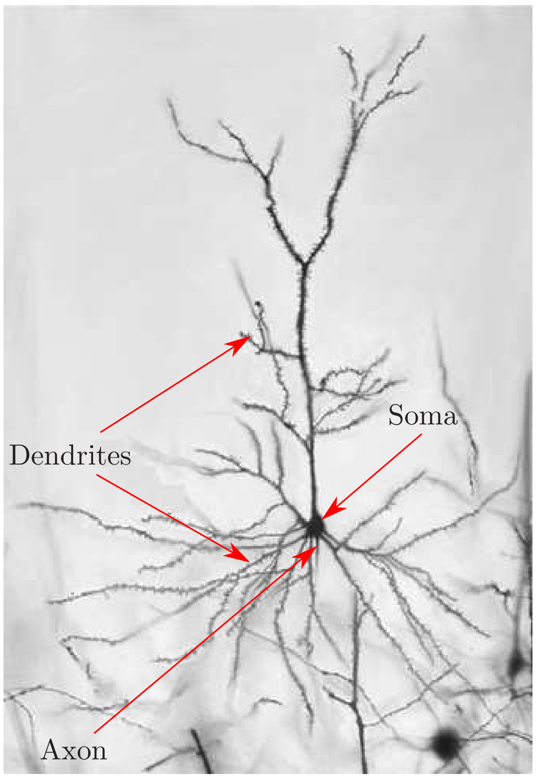


Figure 1.1: Photograph of a stained cortical neuron from rat. The soma, the dendrites and the axon are indicated. Courtesy of Grazyna Gorny.

total cable length of a typical pyramidal neuron are in the order of centimeters (Dayan and Abbott, 2001), while the synaptic connections that are spread across these branches have an area diameter of approximately 1 µm per contact (Shepherd, 2004). This structural density on a microscopic level makes it extremely difficult to access information about the precise interconnection of the cells. Furthermore, the massive number of brain constituents – the human cortex is assumed to comprise approximately 10^{10} cells and 10^{14} synapses (Shepherd, 2004) – introduces an additional obstacle on the way towards an understanding of the relation between structure and function.

Wiring and Firing

In order to understand computational principles of the brain, both its “hardware”, i.e. the morphological aspects of its architecture, and its “software”, the neuronal code that is used to represent, process and store information on the basis of spatio-temporal spike patterns (Gerstner and Kistler, 2002), need to be investigated.

In the early 20th century, first staining techniques developed by Camillo Golgi (for a review see Torres-Fernández et al., 2006) revealed important insights into the morphology of single cells and network anatomy, e.g. the layered laminar structure of the cortex (Ramon y Cajal, 1911, reviewed in Douglas and Martin, 2004). Since then, staining and microscopical techniques have significantly improved, and today even the automated identification and tracing of a single axon and dendrite in volumes of dense neural tissue with electron microscopy has become possible (Briggman and Denk, 2006). This has not yet been systematically performed for full vertebrate brains. Even if the complete static morphology of a brain was available, it would lack important information about the dynamical properties of its constituents.

Therefore, measuring the ongoing activity in neural networks *in vivo* and *in vitro* is inevitable and approached with various techniques. An early non-invasive method is the electroencephalography (EEG, Jung et al., 1979), which today is accompanied by magnetoencephalography (MEG, Cohen, 1968; Waldert et al., 2008) and functional magnetic resonance imaging (fMRI, Logothetis et al., 2001). The spatial resolution of fMRI is in the millimeter order, which cannot be achieved with EEG and MEG. The temporal resolution of fMRI is only in the order of two seconds, though, while EEG and MEG can resolve neuronal activity at the sub-millisecond scale. Another possibility to acquire three-dimensional activity-dependent images of brain regions is the positron emission tomography, with a spatial resolution on the millimeter scale and sampling rates of up to 60 Hz (see e.g. Purschke et al., 2005; Langner, 2003).

Invasive methods provide both a high spatial a high temporal resolution, but usually access only tiny fractions of neural tissue at a time. Extracellular recordings reveal local field potentials generated by volumes with diameters in the order of 100 µm (Mehring et al., 2003), and intracellular recordings with patch-clamp techniques provide highly resolved current flow and voltage measurements across patches of single neuron membranes (Sakmann and Neher, 1995). By recording neighboring neurons with patch-clamping techniques, the correlation between their membrane potential fluctuations can be determined (Lampl et al., 1999; Okun and Lampl, 2008). Thereby, recording pairs or groups of neurons can reveal the existence of synaptic connections and effects of synaptic plasticity, i.e. the changing impact that a neuron has on another neuron. Long-term changes in such synaptic weights have been observed by correlating the pre-synaptic spiking activity with the post-synaptic spike and membrane recordings (Bi and Poo, 1997; Dan and Poo, 2004; Markram et al., 1997). Short-term synaptic plasticity effects have been investigated with similar techniques (Markram et al., 1998; Zucker and Regehr, 2002).

Voltage-sensitive dyes allow for the temporally and spatially highly resolved recording of hundreds of neighboring neurons at the same time (Jin et al., 2002). The possible volume depth can be increased up to a millimeter by applying multi-photon imaging techniques (Kerr et al., 2005; Xu et al., 1996).

In spite of these and many more highly sophisticated methods, a lot of important questions are left unanswered, e.g. regarding the encoding of information in brains, the neural representation of concepts, the creation and the persistence of memory, the robustness of functionality, the ability to learn and the emergence of creativity, to just name a few.

1.2.2 Insights through Modeling

The experimental investigation of neural tissue is inevitable for the generation of a detailed brain knowledge base from which understanding of underlying principles can emerge. The above sketch of neuroscientific measurement techniques indicates the technological difficulties of this endeavor. There are further disadvantages in the experimental study of biological neural systems, for example the following:

- It is hard to control the experimental conditions during *in vivo* studies.
- *In vitro* preparations do not reflect all aspects of living tissue.
- It is difficult to acquire data with large statistics.
- Ethical issues and animal rights need to be considered.
- Most efforts imply high financial costs.

Models are an essential scientific instrument to describe a system in a problem-specific context. Starting from well defined questions, existing pieces of knowledge, methods and hypotheses are combined and evaluated in order to extract the desired answers and predictions. Especially in research fields on dynamical systems that cannot be fully conceived by experimental methods, like astrophysics and neuroscience, models represent an inevitable approach to test hypotheses and theories.

1.2.3 Software Simulators vs. Neuromorphic Hardware

Models of spiking neurons are normally formulated as sets of differential equations for an analytical treatment or for numerical simulation. In contrast to *in vivo* or *in vitro* studies, a software model offers access to all desired observables at any time. It is arbitrarily flexible in its structure, in its level of detail and in the choice of parameters. There is a large community of scientists who contribute significant insights into neuroscience by applying models that usually draw on experimental data, but abstract and transfer these findings into synthetic systems.

For example, varying integrative properties of neuron membranes have been observed *in vivo* (reviewed in Destexhe et al., 2003) and could be well reproduced and explained by corresponding models (Shelley et al., 2002; Rudolph and Destexhe, 2003; Kumar et al., 2008). *In vivo* evidence for long-term synaptic plasticity (Levy and Steward, 1983; Bi and Poo, 1997; Markram et al., 1997; Dan and Poo, 2004) has led to various mathematical descriptions (see e.g. Bienenstock et al., 1988; Song et al., 2000; Legenstein et al., 2005; Morrison et al., 2007, 2008). Models on various spatial scales and with different trade-offs between constituent detail and network size exist, ranging from highly complex single cell descriptions (Jolivet et al., 2008) to large-scale models of significant fractions of cortical tissue (EPFL and IBM, 2008; Johansson and Lansner, 2007). Considered periods range from short-term plasticity effects on the scale of tens of milliseconds (e.g. Tsodyks and Markram, 1997) to structural synaptic development on the scale of hours and days (e.g. Helias et al., 2008).

But the more complex the underlying neuron model and the size and connectivity of the simulated network grow, the more critical computation time gets. If it comes to large networks, statistics-intensive analyses or long-term observations of network dynamics can become computationally extremely expensive (see e.g. Morrison et al., 2005, 2007). The main bottleneck is the mapping of the intrinsic parallelism of neural computation to a relatively small number of sequentially operating processors.

Neuromorphic hardware systems represent an alternative approach that can overcome some of the limitations inherent to pure software simulations (for a review see Renaud et al., 2007). In a physical, typically silicon form they mimic the structure and emulate the function of biological neural networks. Neuromorphic hardware engineering has a tradition going back to the 1980s (Mead and Mahowald, 1988; Mead, 1989), and today an active community is developing analog or mixed-signal VLSI models of neural systems.

Inspired by Mead, silicon retinas (Serrano-Gotarredona et al., 2006) and neuromorphic visual processing systems (Merolla and Boahen, 2006) are still in focus of ongoing research, with various applications such as light-weight and power-efficient sensors in flying robots (Netter and Franceschini, 2002), substitutes of biological visual systems useful e.g. for lecture demonstrations (Delbrück and Liu, 2004) or safety sensors in elderly care (Fu et al., 2008). Devices have been developed that serve as self-tuning motor control units for robotics (Lewis et al., 2000). Others resemble hippocampal place cells as attractor memory devices (Vogelstein et al., 2007) or implement self-organizing spike pattern classifiers (Häfliger, 2007; Mitra et al., 2009).

A more immediate contact with neuroscientific research is achieved e.g. by hybrid setups that couple neuromorphic hardware devices with living tissue (Bontorin et al., 2007). Recent developments, which are in focus of this thesis, aim at the utilization of neuromorphic systems as flexible modeling tools to approach neuroscientific questions (Schemmel et al., 2007; Brüderle et al., 2007; Schemmel et al., 2008; Ehrlich et al., 2007).

Pros and Cons of Neuromorphic Models

The main advantage of the physical emulation of neural network models, compared to their numerical simulation, arises from the locally analog and massively parallel nature of the computations. This leads to neuromorphic network models being typically highly scalable and being able to emulate neural networks in real time or much faster, independent of the underlying network size. Often, only the inter-chip event-communication bandwidth sets a practical limit on the scaling of network sizes by inter-connecting multiple neural network modules (Costas-Santos et al., 2007; Berge and Häfliger, 2007; Schemmel et al., 2008). Compared to numerical solvers of differential equations which require Von-Neumann-like computer environments (Brette et al., 2006), neuromorphic models have much more potential for being realized as miniature embedded systems with low power consumption.

A disadvantage is the limited flexibility of the implemented models. Typically, neuron and synapse parameters and the network connectivity can be programmed to a certain degree within limited ranges by controlling software. However, changes to the implemented model itself usually require a hardware re-design, followed by production and testing phases. This process normally takes several months.

Unlike most numerical simulations of neural network models, analog VLSI circuits operate in the continuous time regime. This avoids possible temporal discretization artifacts, but also makes it impossible to interrupt an experiment at an arbitrary point in time and restart from an identical, frozen network state (see Section 2.1.2 for the definition of an *experiment run* on such a system).

Furthermore, it is not possible to perfectly reproduce an experiment because the device is subject to noise (see Section 4.2.2), to cross-talk from internal or external signals (see Section 4.3.8), and to temperature dependencies (see Dally and Poulton, 1998). These phenomena often have a counterpart in the biological specimen, but it is highly desirable to control them as much as possible.

Another major difference between software and hardware models is the finiteness of any silicon substrate. This in principle also limits the size of any software model, as it utilizes standard computers with limited memory and processor resources, but for neuromorphic hardware the constraints are much more immediate: The number of available neurons and the number of synapses per neuron have strict limits; the number of manipulable parameters and the ranges of available values are fixed.

Exploiting Speed and Scalability

Still, neuromorphic network models are highly scalable at constant speed due to the intrinsic parallelism of their circuit operation. This scalability results in a relative speedup compared to software simulations, which gets more and more relevant the larger the simulated networks become, and which provides new experimental possibilities. A hardware experiment can be repeated many times within a short period, allowing the common problem of insufficient statistics due to lacking computational power to be overcome. Large parameter spaces can be swept to find an optimal working point for a specific network architecture, possibly narrowing the space down to an interesting region which can then be investigated using a software simulator with higher precision. One might also think of longer experiments than have so far been attempted, especially long-term learning tasks which exploit synaptic plasticity mechanisms (Schemmel et al., 2007).

Except for the kind of systems considered in this thesis, all neuromorphic hardware projects cited above currently work with circuits operating in biological real-time. This allows interfacing real-world devices such as sensors (Serrano-Gotarredona et al., 2006) or motor controls for robotics, as well as setting up hybrid systems with *in vitro* neural networks (Bontorin et al., 2007). In contrast to these real-time systems, the type of neuromorphic hardware in focus of the presented work (Schemmel et al., 2007, 2008) operates at a highly accelerated rate (see Section 3.1.5). This crucial feature even increases the speedup advantages mentioned above by many orders of magnitude and hence opens up new prospects and suggests new experimental paradigms (see Section 3.1.1).

1.2.4 Requirements for the Establishment of Neuromorphic Modeling

The computation speed, together with an implementation path towards architectures with low power consumption and very large scale networks (Schemmel et al., 2008; Fieres et al., 2008), makes neuromorphic hardware systems a potentially valuable research tool for the modeling community, where software simulators are more commonplace (see (Brette et al., 2006) for a review of simulation tools). The establishment of a neuromorphic hardware device as a useful component within the neuroscientific modelers' toolbox requires:

- A proof of its biological relevance, i.e. it has to be verified that the implemented neuron and connectivity model can be used to generate biologically realistic structure and behavior.

- For this purpose, a common concept of experiment description and output data interpretation has to be found which allows a comparison between the hardware domain and a reference system. A very practical choice for such a reference are established software simulators.
- Its operability by non-hardware-experts.

In the following chapters, the fulfillment of these requirements for one specific type of neuromorphic device is documented.

1.3 Utilized Neuroscientific Concepts

In the following, a selection of neuroscientific concepts is described that are relevant in the context of this thesis. In case of the synapse modeling, the list does not imply a qualitative superiority of the selected approaches compared to the manifold possible alternatives, but rather reflects the functionality that is implemented in the hardware system utilized in the following chapters (see Section 2.1).

1.3.1 High-Conductance States

Membrane dynamics of single neurons play an important role in neural information processing. Activity measurements in the cortex show that the dynamical properties of a membrane are strongly influenced by the total of its synaptically induced conductances (Shu et al., 2003; Destexhe et al., 2003; Boustani et al., 2007; Cossart et al., 2003). In this context, it is useful to distinguish between two states of neuronal activation: *Up* states or *activated* states, where the membrane is depolarized by increased extracellular activity and the embedded cell fires irregularly, and *down* states, where both intra- and extracellular activity follow low-frequency rhythms (Anderson et al., 2000). In the activated state, which is also called the *high-conductance state*, neurons show stochastic firing behavior and an enhanced responsiveness towards input stimuli.

The high-conductance state determines the properties of a single neuron's membrane within an active network. There is experimental evidence for its existence within *in vivo* networks, e.g. in awake and attentive animals (Destexhe et al., 2003; Boustani et al., 2007), or *in vitro* in localized sub-populations (Cossart et al., 2003).

Characteristics of neurons in the high-conductance state are a low input resistance, a depolarized membrane with large membrane potential fluctuations, dominant inhibitory conductances and a stochastical response to stimulation patterns due to fluctuating background activity (Destexhe et al., 2003; Kumar et al., 2008).

In (Wielaard et al., 2001) and in (Shelley et al., 2002), a model is described and analyzed which is based on leaky integrate-and-fire neurons that can exhibit high-conductance states. There, the membrane potential $V(t)$ (see also Equation 2.1) is shown to follow the so-called *effective reversal potential* $V_{\mathrm{eff}}(t)$ with the membrane time constant $\tau_{\mathrm{m}}(t)$. $V_{\mathrm{eff}}(t)$ is defined as the *difference current* $I_{\mathrm{D}}(t)$ between excitation and inhibition divided by the total membrane conductance $g_{\mathrm{T}}(t)$:

$$V_{\mathrm{eff}}(t) \equiv \frac{I_{\mathrm{D}}(t)}{g_{\mathrm{T}}(t)} = \frac{g_{\mathrm{e}}^{\mathrm{tot}}(t)E_{\mathrm{e}} - g_{\mathrm{i}}^{\mathrm{tot}}(t)|E_{\mathrm{i}}|}{g_{\mathrm{T}}(t)} \quad . \tag{1.1}$$

Here, $g_{\mathrm{e}}^{\mathrm{tot}}(t)$ and $g_{\mathrm{i}}^{\mathrm{tot}}(t)$ denote the total of synaptically induced excitatory and inhibitory conductances, respectively. The membrane time constant $\tau_{\mathrm{m}}(t)$ is determined by the total membrane capacitance C_{m} and the total conductance $g_{\mathrm{T}}(t) \equiv g_{\mathrm{l}} + g_{\mathrm{e}}^{\mathrm{tot}}(t) + g_{\mathrm{i}}^{\mathrm{tot}}(t)$, where g_{l} models the permanent and constant leakage conductance (see also Equation 2.1):

$$\tau_{\mathrm{m}}(t) \equiv \frac{C_{\mathrm{m}}}{g_{\mathrm{l}} + g_{\mathrm{e}}^{\mathrm{tot}}(t) + g_{\mathrm{i}}^{\mathrm{tot}}(t)} \qquad . \tag{1.2}$$

This membrane time constant can be understood as the temporal resolution capability of the neuron, because a small $\tau_{\mathrm{m}}(t)$ makes the membrane potential immediately follow the effective reversal potential and therewith follow the synaptic input. Consequently, a neuron with a small $\tau_{\mathrm{m}}(t)$ can perform well as a coincidence detector because it is able to rapidly detect changes in input correlation.

The possibility of neuron membranes to switch between integrating properties and coincidence detector functionality enriches the information processing capabilities of neural networks.

1.3.2 Models of Synapse Response Dynamics

In the following, some popular models for the temporal development of synaptic currents or conductances as a response to incoming spikes are listed. In this thesis, they are also referred to as *temporal kernels* or, only in the case of conductance-based synapses, *conductance courses* (CC).

Delta Function For current-based models, the synaptic response to a spike arriving at time $t = t_{\mathrm{sp}}$ is sometimes modeled simply as current delta peaks, see e.g. (Brunel, 2000):

$$I_{\mathrm{syn}}(t) = I_0\, \delta(t - t_{\mathrm{sp}}) \qquad . \tag{1.3}$$

Quantal Increase and Exponential Decay A modification which introduces temporal dynamics to the synapse is a quantal increase of the conductance or current by a fixed value w_{syn} (usually referred to as the *synaptic weight*), followed by an exponential decay with time constant τ_{syn}, found e.g. in (Sussillo et al., 2007) and (Maass et al., 2004a):

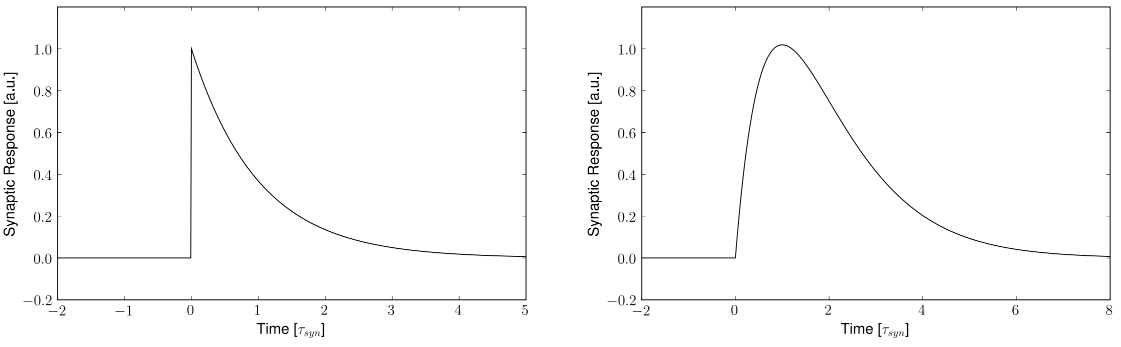
$$\frac{\mathrm{d}g_{\mathrm{syn}}(t)}{\mathrm{d}t} = -\frac{g_{\mathrm{syn}}(t)}{\tau_{\mathrm{syn}}} + w_{\mathrm{syn}}\, \delta(t - t_{\mathrm{sp}}) \qquad . \tag{1.4}$$

See Figure 1.2(a) for an illustration.

Alpha Function In order to avoid the quantal increases of currents or conductances and rather have a non-instantaneous rise which smoothly changes into a decrease, alpha or alpha-like functions are commonly used, e.g. in (Kumar et al., 2008) and (Shelley et al., 2002):

$$g_{\mathrm{syn}}(t) = g_0\, \frac{t - t_{\mathrm{sp}}}{\tau_{\mathrm{syn}}} \exp^{(t_{\mathrm{sp}} - t)/\tau_{\mathrm{syn}}}\, H(t - t_{\mathrm{sp}}) \qquad , \tag{1.5}$$

where $H(t)$ is the Heaviside step function. See Figure 1.2(b) for an illustration.



(a) Quantal increase and exponential decay

(b) Alpha function

Figure 1.2: Two typical examples for the temporal response kernels of modeled synapses. Such kernels are used to shape the conductance or the current course generated when a spike arrives at the synapse. (a) A quantal increase followed by an exponential decay. (b) An alpha function.

1.3.3 Synaptic Learning

In the following, two important models of synaptic plasticity are introduced: The short-term mechanisms of synaptic depression and facilitation, and a model of long-term synaptic modification based on the temporal correlation of pre- and post-synaptic spike times. For a review of phenomenological models of synaptic plasticity based on spike timing, see e.g. Morrison et al., 2008.

Short-Term Synaptic Plasticity

It has been found that the efficacy of biological synapses can be dependent on the history of their pre-synaptic activity (Tsodyks and Markram, 1997; Markram et al., 1998). These changes typically last for a few milliseconds to seconds. The effect is called *depression* if the synapse gets weaker, the opposite effect, i.e. strengthening of the synapse, is referred to as *facilitation*. See Figure 1.3 for a schematic.

In the FACETS Stage 1 hardware, a slight modification of the short-term plasticity model described in Tsodyks and Markram, 1997 is implemented. It is introduced in Section 2.1.3,

Long-Term Synaptic Plasticity

A first important postulation about the mechanisms of self-organization in the brain was made by the psychologist Donald Hebb in 1949, especially notable since it was formulated on purely theoretical grounds (Hebb, 1949). He suggested a dependency of the development of synaptic strengths on correlations between pre- and post-synaptic activity: “When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.”

Based on experimental evidence described e.g. in Bi and Poo, 1997 and (Markram et al., 1997), a class of synaptic weight modification models has been developed that is based on the temporal correlation between pre- and post-synaptic firing (Song et al., 2000; Legenstein et al., 2005; Morrison et al., 2007, for a review see Morrison et al., 2008). According to these

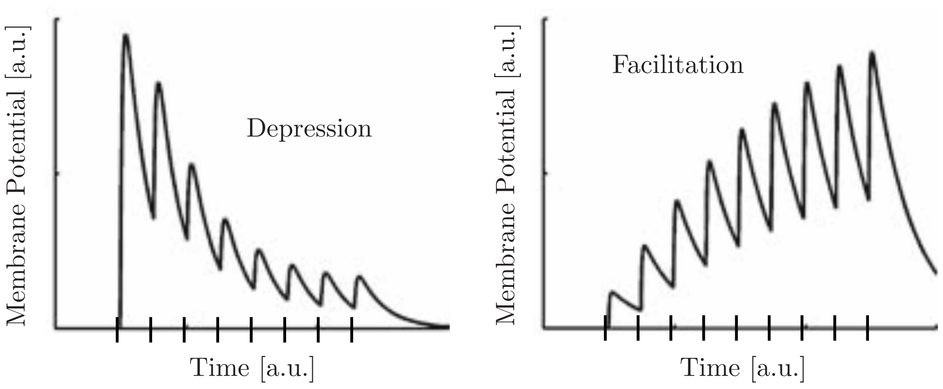


Figure 1.3: Schematic of a membrane potential that receives sequences of spikes (indicated as vertical markers on the time axis) via an excitatory depressing (left) and via an excitatory facilitating (right) synapse. In the case of depression, the synaptic efficacy decreases due to the successive pre-synaptic input. In the case of facilitation, the synaptic efficacy grows as a consequence of the pre-synaptic stimuli.

so-called *spike-timing-dependent plasticity* (STDP) rules, the weight of a synapse increases by an additive or relative amount, if the temporal correlations between pre- and post-synaptic spikes suggest a causal relationship. In cases of acausal correlations, the weight will decrease. Such correlations are determined in terms of spike pairings or triplets (Pfister and Gerstner, 2006).

For the specific STDP implementation realized in the FACETS Stage 1 hardware system, see Section 2.1.3.

1.4 The FACETS Research Project

The FACETS[2] research project (FACETS, 2009) is a collaboration of 16 European partner groups, funded by the European Commission in the framework of the Information Society Technologies (IST, 2009) program. FACETS puts together a multi-disciplinary team of researchers from the fields of biology, mathematics, computer science, physics and electrical engineering. The goal of the project is to experimentally and theoretically investigate brain-like computing principles beyond the Turing paradigm (Turing, 1937; von Neumann, 1945). Computational concepts inherent to biological neural systems, which form the basis for capabilities like the processing of sensory input, memory, self-organization and learning, shall be extracted, understood, generalized and exploited for technological applications.

The research in FACETS follows three major approaches:

- *In vivo and in vitro experiments with real neural tissue.*
  The acquired results are collected in data bases and therewith made accessible to other FACETS members.

- *The theoretical analysis of the experimental data.*
  Mathematical models of neural systems based on the experimental observations are

[2]Fast Analog Computing with Emergent Transient States

extracted. If possible, they are investigated analytically, otherwise they are numerically computed in software simulations.

- *The design, implementation and utilization of neuromorphic hardware devices which physically implement the developed neural network models.*

The mutually beneficial cooperation of researchers working in these three fields is a major challenge aimed at and strongly supported by FACETS.

Neuromorphic Hardware in FACETS Along the line of keeping diversity instead of eliminating it, the project follows two development branches for neuromorphic hardware systems: One approach aims at VLSI models of neurons with a high degree of biological precision. These systems are operating in real-time and provide the opportunity to set up hybrid systems, i.e. networks of biological *and* hardware neurons which communicate with each other (Bontorin et al., 2007). The second approach tries to exploit the possibilities of CMOS[3] technology in terms of the small realizable time scales, the low power consumption and the possible dense spatial integration of VLSI circuits. The goal is to create a novel, large-scale and massively accelerated neuromorphic hardware device which, among other possible applications (see Section 3.1.1), provides a complementary modeling tool for the computational neuroscience community. It shall helps to overcome some typical limitations of software simulators like speed and scalability (see Section 1.2.3). In Chapter 2, this second type of hardware is introduced and described in detail.

The development of such a device is not a task performed isolatedly by the engineering groups within FACETS. Instead, its successful realization essentially depends on contributions from modeling experts and experimentalists. The implemented neuron and synapse models are the outcome of project-wide discussions, and the communication structure provided by the currently developed large-scale system (see Section 2.2), for example, strongly orients towards the realization of experiment types performed and suggested by FACETS members.

[3]Complementary Metal Oxide Semiconductor

2 Neuromorphic Substrate

This chapter introduces the two development stages of the accelerated, large-scale hardware approach followed by the FACETS project (see Section 1.4). Section 2.1 provides a detailed description of the chip-based prototype devices, also referred to as `FHW-1` systems. These devices have been available for some time already. They have been utilized for all experimental hardware work presented in this thesis. Many of the `FHW-1` components and sub-circuits will be used in the target system, which is not based on single chips anymore, but on the creation of a neural substrate on a whole silicon wafer. This *wafer-scale* or `FHW-2` system, which is currently under development, is introduced in Section 2.2.

2.1 Chip-Based Neural Network Model

The components of the hardware system utilized throughout this thesis have been described in detail in various publications (Schemmel et al., 2004, 2006, 2007; Fieres et al., 2004; Philipp et al., 2007; Grübl, 2007; Philipp, 2008). In the following, basic information about the neural network chip and its support hardware is summarized with respect to the neuroscientific applicability of the system. The implemented neuron and synapse model plus the realizable network topologies are described.

Furthermore, since the correspondence between hardware parameters and their biological counterparts is not always a linear one (see Section 3.1.5), and since multiple process-inherent or design-related hardware mechanisms impose problems on the biologically realistic operation of the device (see Section 4.3), a selection of technical details about sub-circuits, about hardware parameters and about their precise functionality is provided. Later sections will refer to these details.

2.1.1 Technology and Dimensions

The existing first three versions of the FACETS Stage 1 hardware (`FHW-1`) system are built using a standard 180 nm CMOS process. 384 neurons are located on one single 25 mm^2 die. The programmable inter-neuron connectivity is described below in Section 2.1.2. The possibility to inter-connect multiple of the network dies is work in progress (Philipp et al., 2007). The system exhibits an acceleration factor of up to 10^5 – see Section 3.1.5 for a detailed explanation of the applied time transformation method. The action potentials generated during hardware operation can be recorded with a temporal resolution of approximately 0.3 ns (`HTD`). The precise value of this depends on the frequency f_{chip} the chip is clocked with.

Figure 2.1 shows a photograph of an `FHW-1.2` chip bonded to a carrier PCB[1]. The two symmetric, homogeneously structured rectangles are the synapse arrays described below in Section 2.1.2. The more heterogeneously structured rectangle below the synapse arrays comprises the neuron circuits described in Section 2.1.2 and various digital control circuitry.

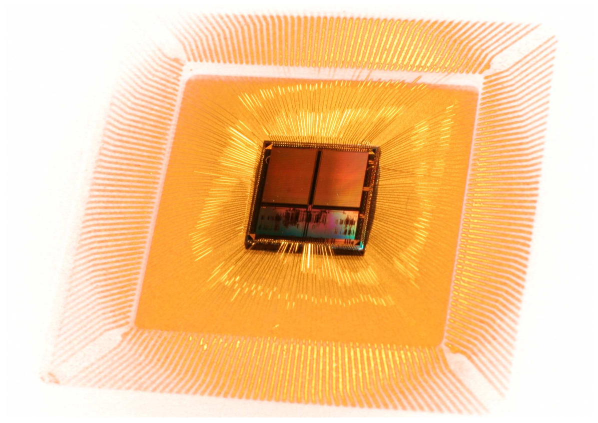


Figure 2.1: Photograph of an `FHW-1` chip and the bonding wires connecting it to pads inside a package.

2.1.2 Implemented Model

The `FHW-1` system implements a leaky integrate-and-fire (I&F) neuron model with conductance-based synapses (Destexhe, 1997; Destexhe and Pare, 1999). It is designed to exhibit a linear correspondence with existing phenomenological conductance-based modeling approaches (Burkitt et al., 2003; Rudolph and Destexhe, 2006). The cell body is modeled as a *point neuron*, i.e. it does not comprise any compartments separated by resistors which are often used to model the temporal dynamics induced by the spatial extension of the cell soma, its dendrites and its axon (Gerstner and Kistler, 2002, Section I.4.4).

In analytical or numerical I&F neuron models, the electrical processes of neural information processing are usually described mathematically by sets of differential equations. In the neuromorphic hardware described here, nearly all electric quantities incorporated in the model are physically represented: A capacitance is modeled by a capacitance, currents are currents and voltages are voltages. Conductances are emulated by voltage controlled current sources, so-called operational transconductance amplifiers. The model is not passively computed by a separate unit, *it computes itself* in continuous time and with continuous variables.

What is an Experiment?

An important consequence of the physical nature of the neuromorphic model is the hardware-specific definition of an *experiment run*: In contrast to pure software simulations, where time

[1]Printed Circuit Board

emerges as a sequence of discrete numerical integration steps, the analog part of a neuromorphic hardware model exists and operates continuously in real-time. In software simulations, an experiment run has a starting point in time with well defined initial conditions plus an experiment duration. If not explicitly computed, the model does not exist beyond this scope. In a hardware system connected to its power supply, the membrane potential is permanently existing and evolving and can always be measured.

Hence, an experiment run in hardware is defined as a period T_{exp} during which

- the system is fully configured with well defined parameter values,
- a sequence of stimuli is applied and
- a set of observables is recorded.

A period T_{init} is recommended which immediately precedes T_{exp} without stimulation and recording, but with a well defined hardware configuration already applied. This minimizes the influence of the operation history on dynamically evolving observables like the membrane potential or synaptic activation states. Still, due to effects described in Sections 4.2.2 and 4.3.8, two hardware runs with identical settings can never generate precisely the same results.

Membrane Potential Dynamics

The neuron circuits are designed such that the temporal development of an emulated membrane potential $V(t)$ is determined by the following differential equation:

$$-C_{\text{m}}\frac{dV}{dt} = g_{\text{l}}(V - E_{\text{l}}) + \sum_j p_j(t) g_j^{\text{amp}}(t)(V - E_{\text{e}}) + \sum_k p_k(t) g_k^{\text{amp}}(t)(V - E_{\text{i}}) \quad . \tag{2.1}$$

The constant C_{m} represents the total membrane capacitance. The first term on the right-hand side models the sum of all currents via constantly conducting ion channels. Hence, if no transient conductances towards other reversal potentials are active, the membrane potential will develop towards the so-called *leakage reversal potential* E_{l}. The time constant of this development, an exponential convergence, is determined by the membrane capacitance and the so-called *leakage conductance* g_{l}: $\tau_{\text{m}} = \frac{C_{\text{m}}}{g_{\text{l}}}$.

The transient conductances imposed by synaptic activity have different reversal potentials, E_{i} and E_{e}, modeling inhibitory and excitatory ion channels. The index j in the first sum of Equation 2.1 runs over all excitatory synapses, while the index k in the second sum covers the inhibitory ones. The conductance course (CC) induced by an individual synapse s is modeled as a product $p_s(t)g_s^{\text{amp}}(t)$, where $p_s(t)$ denotes the synaptic open probability (Dayan and Abbott, 2001), and g_s^{amp} is again a product of the synaptic weight $\omega_s(t)$ and a maximum conductance $g_s^{\text{max}}(t)$:

$$g_s^{\text{amp}}(t) = \omega_s(t)\, g_s^{\text{max}}(t) \quad . \tag{2.2}$$

The shape of a CC is determined by $p_s(t)$ (see Section 2.1.2), while $g_s^{\text{amp}}(t)$ defines its amplitude. The weights $\omega_s(t)$ can be dynamically modified by the implemented so-called *spike-timing dependent plasticity* (STDP, see Section 1.3.3) algorithm which in every synapse detects temporal correlations in the order of tens of milliseconds (BVD, Caporale and Dan, 2008) between pre- and post-synaptic action potentials. See Section 2.1.3 for more detail on that. The maximum conductances $g_s^{\text{max}}(t)$, on the other hand, can be affected by plasticity mechanisms called *short-term depression* and *short-term facilitation*, both determined by the pre-synaptic activity only.

The neuron emits a spike as soon as a configurable threshold voltage V_{thresh} is exceeded. In the FHW-1 neuron model, *emitting a spike* means that a circuit separate from the neuron membrane releases a short voltage pulse, which is delivered to the synapse circuits of possibly connected target neurons. This pulse can also be recorded by the operating software, then comprising the neuron index and an event time stamp, discretized (Grübl, 2007) with a temporal resolution of approximately 0.3 ns (HTD). A strong depolarization peak on the membrane like in biological neurons is not modeled in hardware at all, because it is assumed to carry no relevant information except for the time of its occurrence. Once a spike has been released, the hardware membrane potential is forced to a reset voltage V_{reset}, where it remains for a so-called *refractory period* τ_{ref}, then it is released back into the influence of excitatory, inhibitory and leakage mechanisms. Figure 2.2 exemplarily shows such a spike situation recorded from an FHW-1 neuron.

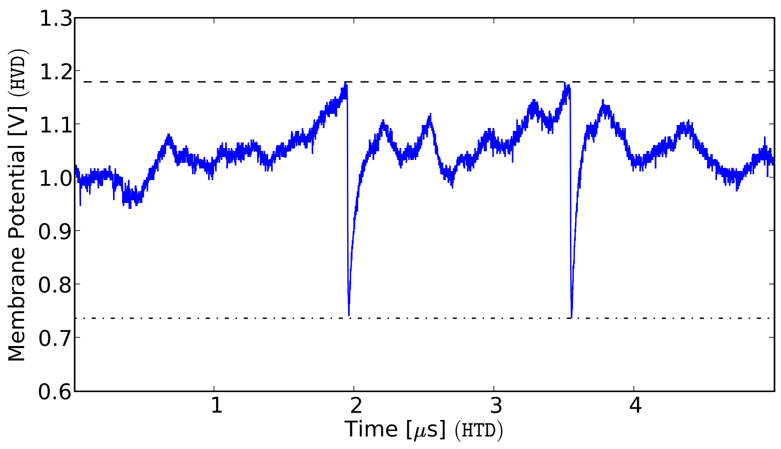


Figure 2.2: Detail of a membrane potential recorded from an FHW-1 neuron that is stimulated with excitatory and inhibitory Poisson-type input. Time and voltage are given in hardware time domain (HTD) units. Two times, at approximately $t_1 = 2\,\mu\text{s}$ and $t_2 = 3.5\,\mu\text{s}$, the membrane voltage exceeds the firing threshold V_{thresh}, which is indicated by the dashed line. Then, as explained in the main text, a separate circuit releases a spike signal, while the neuron membrane is pulled to its reset potential V_{reset}, indicated by the dash-dotted line.

Connectivity and Synapse Model

In the following, the programmable FHW-1 inter-neuron connection structure and the synaptic conductance dynamics are described.

Programmable Connections The 384 neurons are located in two so-called *network blocks* of 192 circuits each. Figure 2.3 illustrates the implemented connectivity mechanisms within and between these blocks. Each network block contains 256 *synapse drivers* (in the figure: triangles) and $256 \cdot 192 = 49152$ *synapse nodes* (circles) which can be individually programmed to connect a synapse driver to a neuron (squares) via a certain weight. Every synapse driver is connected to 192 synapse nodes and can be programmed to receive one of three possible signals: (a) The output spikes of one specific neuron located in the same network block, (b) the output spikes of one specific neuron located in the other neuron block, or (c) events which have been generated externally and which are delivered via software and digital control logic within the support hardware (see Section 2.1.5) and within the chip itself. The signal source can be programmed individually for each synapse driver and, by means of the current operating framework, is not changeable during one experiment run. Furthermore, pairs of neighboring synapse drivers can be configured such that they share the same input source, i.e. that pairs of synapse node rows can be combined to one virtual row with an effectively increased weight resolution per virtual node. The spike output of every neuron can be fed to either one specific synapse driver in the same network block, to the corresponding driver in the opposite block, to some digital recording logic, or to every possible combination of those three targets.

Consequently, if the neuron blocks are used isolatedly, each neuron can be connected to every other neuron in the same block. In such a full connection scheme, 64 synapse drivers remain in each network block for the application of externally generated spikes. But if both network blocks are combined to form one larger network with feedback connections, configuration conflicts can emerge. The operation software has to detect and minimize such conflicts (see Sections 3.2.2 and 3.1.6).

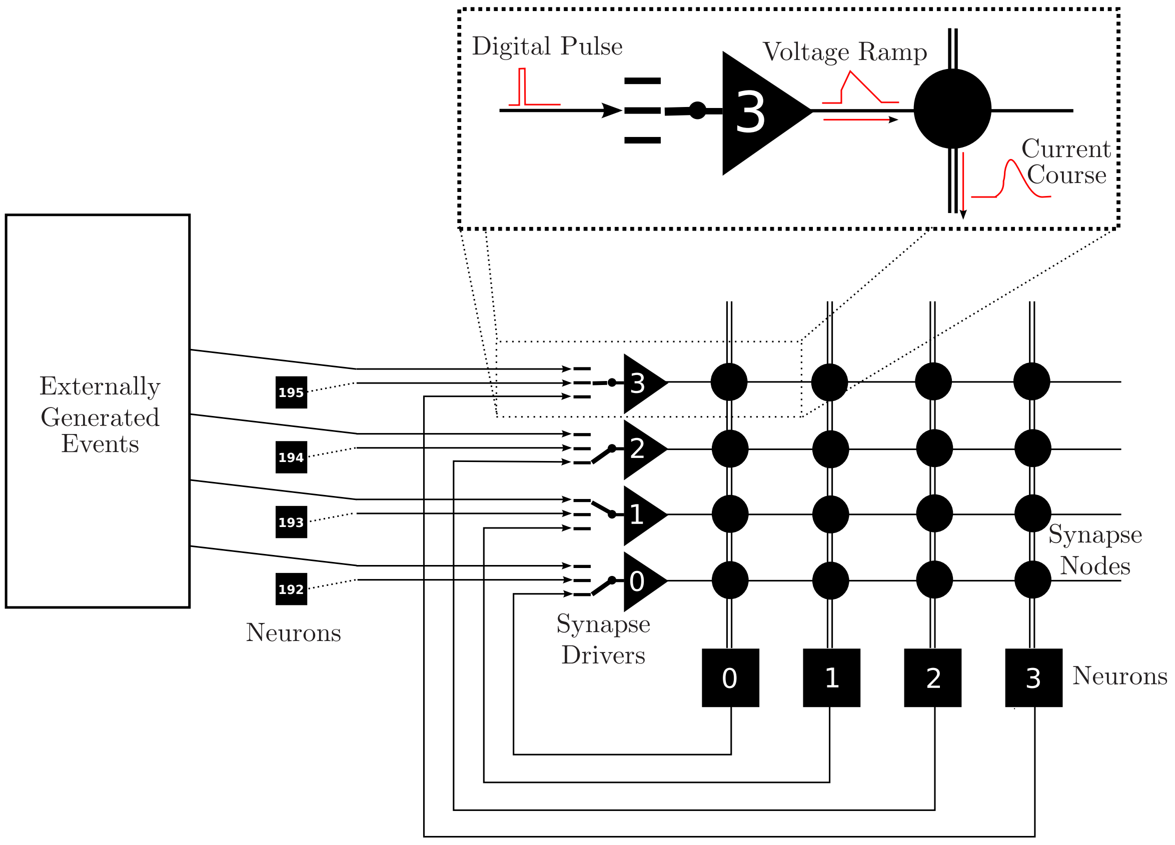


Figure 2.3: Schematic of the connectivity structure on an `FHW-1` chip. The chip consists of two network blocks with 192 neurons each – only a small subset is drawn here. At the synapse drivers (triangles) of a network block, spikes arrive as short analog voltage pulses from three possible sources: From the neuron circuits within the same network block (large squares), from the neuron circuits of the opposite network block (small squares) or from a memory that stores externally generated events. Each synapse driver can be individually connected to exactly one of those three senders. The zoom box illustrates how the arriving spikes are transformed into a synaptically generated conductance course at the neuron membrane: The synapse driver generates a linearly rising and falling voltage ramp, which is delivered to all synapse nodes (circles) within one synapse driver row. Depending on their programmed weights, these nodes inject current courses into the vertical lines that are connected to the neuron circuits. There, these currents control conductances between the neuron membrane and its excitatory or inhibitory reversal potential.

Synapse Dynamics In the `FHW-1` system, conductance-based synapses are implemented. Compared to purely current-based approaches, they offer a voltage-dependent and hence a more realistic impact on their post-synaptic membrane potentials. In Section 1.3.2, some commonly used modeling approaches for the temporal kernels of synaptic conductance courses have been introduced. The synapse dynamics implemented in the hardware system are very similar to the alpha function shape or – if configured adequately – to the quantal increase followed by an exponential decay.

The zoom box in Figure 2.3 illustrates the process of synaptic influence on a hardware membrane potential: The synapse drivers receive action potentials in the form of short voltage pulses. Triggered by these, they generate linearly rising and falling voltage ramps, which are delivered to the connected synapse nodes. See Figure 2.5 for a diagram of the synapse driver circuits. In the synapse nodes, the voltage ramps cause exponentially rising and falling currents, which can be scaled by a four-bit value corresponding to $\omega_s(t)$ (see Equation 2.2). The shape of this current course and the part of its amplitude which is determined by the synapse driver voltage amplitude correspond to $p_s(t)$ and $g_s^{\max}(t)$, respectively (see Equations 2.1 and 2.2).

The current generated by a synapse node is fed into a line, where it is superposed with currents from other nodes. The sum directly controls a conductance at the connected neuron. Every neuron has an excitatory and an inhibitory reversal potential with one corresponding controllable conductance each. Consequently, two of the current lines described above connect these dynamical conductances of each neuron with its 256 synapse nodes. Every synapse node can be configured to inject its current either into the excitatory or into the inhibitory line. The resulting transient values at the controllable conductances pull the neuron membrane potentials towards their excitatory or to their inhibitory reversal potential, mimicking the influence of opening and closing ion channels in real neurons.

2.1.3 Synaptic Plasticity

The `FHW-1` synapses can be configured to automatically change their efficacies according to a short-term plasticity mechanism (see Section 1.3.3) and to a correlation-based long-term modification rule (see Section 1.3.3). The short-term plasticity feature strongly enriches the possible dynamics of implementable networks, see e.g. Section 6.2.1 for an implemented self-stabilizing network architecture. Together with the high speedup of the system, the synaptic long-term modification mechanism implemented in the `FHW-1` chips bears the opportunity to perform statistics-intensive long-term learning experiments that are not yet possible on conventional software simulation platforms.

STDP

A correlation measurement of pre- and post-synaptic activity is modeled according to the principle of spike-timing dependent plasticity (STDP, Song et al., 2000; Legenstein et al., 2005; Morrison et al., 2007, 2008) in every hardware synapse. It is based on the biological mechanism as described e.g. in (Levy and Steward, 1983; Bi and Poo, 1997; Dan and Poo, 2004). For each occurrence of a post-synaptic action potential, the synapse circuit measures the time Δt that has passed since the last occurrence of a pre-synaptic spike. The same is done separately for each pre- after post-synaptic spike pairing, such time differences being handled as negative values. A positive Δt denotes a possible *causal* correlation, while negative values of Δt are referred to as *acausal*.

The exponentially weighted time difference is called the STDP modification function $F(\Delta t)$ and is defined as follows:

$$F(\Delta t) = \begin{cases} A_+ \exp(-\frac{\Delta t}{\tau_+}) & \text{if } \Delta t > 0 \quad \text{(causal)} \\ -A_- \exp(\frac{\Delta t}{\tau_-}) & \text{if } \Delta t < 0 \quad \text{(acausal)} \end{cases} \quad . \tag{2.3}$$

It is a piecewise function and consists of two branches $F_{\mathrm{a}}(\Delta t)$ and $F_{\mathrm{c}}(\Delta t)$ for the acausal and the causal case. Figure 2.4 schematically shows $F(\Delta t)$, with A_+ and A_- being the maximum and minimum amplitudes of the two branches, which decay with time constants τ_+ and τ_-.

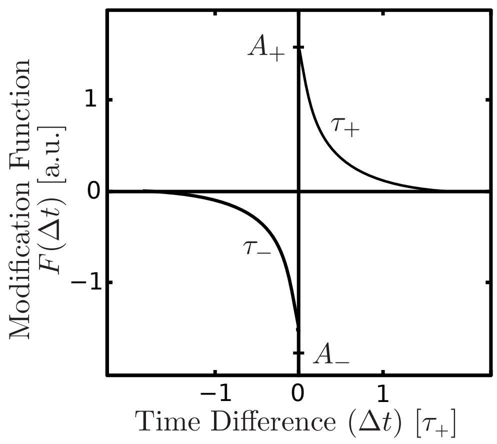


Figure 2.4: Schematic of an STDP modification function. Depending on the time difference $\Delta t = t_{\mathrm{post}} - t_{\mathrm{pre}}$ between a pre- and a post-synaptic spike, the modification function $F(\Delta t)$ of a synapse determines its additive or multiplicative weight change. The function consists of two exponentially decaying branches with the corresponding amplitudes A_+ and A_- and the decay time constants τ_+ and τ_-.

In most STDP models, the modification function represents an additive or multiplicative weight change which is applied to the synaptic weight at each occurrence of a spike pairing or more complex correlation events (Morrison et al., 2008). In the hardware model, the STDP mechanism affects the weights ω_s stored in every synapse, but, due to technical reasons, this cannot be done each time a spike pair has emerged. Since ω_s is represented by four-bit values only, the weight change imposed by a single spike pair is usually below the smallest step determined by this resolution. Hence, the possibly very small values $F(\Delta t_i)$ caused by many succeeding spike pairs P_i have to be stored and accumulated, at least until their sum is large enough to flip a bit in ω_s.

This accumulation is performed in every synapse node by loading two capacitors with charges depending on the individually measured values of $F_{\mathrm{a}}(\Delta t)$ and $F_{\mathrm{c}}(\Delta t)$. The resulting capacitor voltages $V_{\mathrm{a}}^{\mathrm{STDP}}$ and $V_{\mathrm{c}}^{\mathrm{STDP}}$ correspond to the two sums $\sum F_{\mathrm{a}}$ and $\sum F_{\mathrm{c}}$. A digital controller periodically tests for certain conditions: If $|V_{\mathrm{a}}^{\mathrm{STDP}} - V_{\mathrm{c}}^{\mathrm{STDP}}|$ exceeds a configurable correlation threshold $V_{\mathrm{thresh}}^{\mathrm{STDP}}$, the correlation occurrence flag z_{O} will be raised. If in such a case $V_{\mathrm{c}}^{\mathrm{STDP}}$ is greater than $V_{\mathrm{a}}^{\mathrm{STDP}}$, the *causal* correlation flag z_{c}, in the opposite case the *acausal* correlation flag z_{a} will be set.

If z_{O} is set, a weight change will be initiated: In case z_{c} is set, the new value for ω_s is read from a look-up table $\mathrm{LUT}_{\mathrm{c}}$ which holds new weights for every possible old value, i.e. 16 entries altogether. For the acausal case, i.e. if z_a has been raised, a second table $\mathrm{LUT_a}$ provides the new weight to be written. In case both z_c and z_a are set, no weight change is performed. If the digital controller finds a raised z_O and possibly performs the described weight update, it will clear all correlation flags and reset the capacitor voltages $V_\mathrm{a}^\mathrm{STDP}$ and $V_\mathrm{c}^\mathrm{STDP}$ afterwards. If z_O is checked but found to be not set, the digital controller will do nothing.

The frequency with which the correlation flags are checked and the weights possibly manipulated depends on the number of synapses that are defined to be under STDP control. See Section 4.3.11 for more details on this issue. The STDP algorithm can be fully disabled.

Synaptic Depression and Facilitation

A slight modification of the short-term plasticity model described in (Tsodyks and Markram, 1997) is implemented in the `FHW-1` hardware: For every synapse, a time varying so-called *inactive partition* I with $0 \leq I \leq 1$ is electronically represented. I decays exponentially with a recovery time constant τ_rec, but every spike arriving at the synapse increases I by a value depending on the current state of I:

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t} = -\frac{I(t)}{\tau_\mathrm{rec}} + \sum_s C \cdot (1 - I(t)) \cdot \delta(t - t_s) \qquad . \tag{2.4}$$

The sum covers the arrival times t_s of all incoming spikes s. C determines the impact of these spikes on the inactive partition and is a dimensionless value between 0 and 1.

With an adjustable scaling factor $0 \leq \lambda_\mathrm{dep} \leq 1$, depression changes the synaptic conductance amplitude g^max (see Equation 2.2) as follows:

$$g^\mathrm{max}_\mathrm{dep}(t) \propto 1 - \lambda_\mathrm{dep} \cdot I(t) \qquad . \tag{2.5}$$

Given another adjustable scaling factor $0 \leq \lambda_\mathrm{fac} \leq 1$, facilitation increases $g^\mathrm{max}_\mathrm{fac}$ like

$$g^\mathrm{max}_\mathrm{fac}(t) \propto 1 + \lambda_\mathrm{fac} \cdot I(t) \qquad . \tag{2.6}$$

For experiments that illustrate and utilize the short-term plasticity feature in the hardware system, see Sections 6.1.4 and 6.2.2.

2.1.4 Configurability, Ranges and Precision

This section describes a selection of hardware-specific details that will be referred to in later sections, mostly during the analysis of design-related malfunctions and interferences (Section 4.3).

Neuron Membranes For all neuron circuits, the membrane is implemented as a CMOS capacitance that is not adjustable. It is designed to have a value of approximately 200 fF (`HVD`), plus an estimated additional 100 fF more caused by parasitic capacitances of surrounding circuitry. Due to specifications of process variations and experiences with those (see Section 4.2.1), the fluctuation of the produced capacitances from neuron to neuron is assumed to be in the order of 10% (Schemmel, 2008) . The real values after production cannot be determined exactly.

Neuron Voltage Parameters and Refractory Period The reversal potentials E_l, E_e and E_i as well as V_reset and V_thresh are implemented as programmable voltages which are generated on-chip by dedicated circuits (Grübl, 2007, Section 4.5). On every FHW-1 chip, each of the four neuron pools distinguished by the odd and the even indices on the left and on the right network block share one value per voltage parameter. For example, all neurons on the left network block with an even index receive the same value for their firing threshold. Due to circuit differences caused by transistor level variations, this does not necessarily mean that these neurons all respond the same to identical stimulation. Hence, this sharing of parameter values causes problems for the neuron responsiveness calibration (see Section 5.2.5 for an elaboration on that) and for other efforts to achieve an homogeneous substrate.

Input values from 0.0 V to 2.5 V (HVD) can be written to the voltage generator cells with a 10 bit resolution, but these circuits are limited to the generation of output values from approximately 0.6 V to 1.6 V (HVD).

For voltage parameter values typically applied in models of cortical neurons (see Appendix A.1), the distance between the lowest reversal potential and the firing threshold is much smaller than then difference between the threshold and the excitatory reversal potential. As mentioned above, the programmable neuron voltage parameters in hardware are limited to a range of approximately 1.0 V width (HVD). Since action potentials in hardware are only represented digitally and the membranes evolve only in the sub-threshold regime, the voltage range that is actually covered by the membranes is only a fraction of this available range. Furthermore, due to technical constraints imposed by the firing threshold comparator, the voltage V_thresh shall not be set to values larger than 1.1 V (HVD). Hence, in order to avoid a waste of the available dynamic voltage range, i.e. to maximize the signal-to-noise ratio of the membrane potential signal, the possible conductance values between a membrane and its excitatory reversal potential have been designed to be much larger than those towards its inhibitory reversal potential. This design feature allows to set the difference between the excitatory reversal potentials and the firing threshold to approximately the same value as the difference between the firing threshold and the inhibitory reversal potentials, while realistic impacts of synaptic activity onto the membrane potentials can still be achieved for both mechanisms. See Section 3.1.5 for more details on the mapping between biological and hardware voltage values and on the described break of mapping linearity.

Another hardware parameter that turned out to have enormous influence on the membrane dynamics is the bias current $I_\mathrm{thresh}^\mathrm{bias}$ for the firing threshold comparator circuit. The value of $I_\mathrm{thresh}^\mathrm{bias}$ does not only control the speed of threshold crossing detections, but also the period t_reset during which the membrane is connected to its reset potential V_reset with a large conductance g_reset, i.e. the refractory period τ_ref. This double functionality of the parameter $I_\mathrm{thresh}^\mathrm{bias}$ imposes a serious problem, which will be described in Section 4.3.2.

Leakage Conductance The static leakage conductance g_l is controlled by a current $I_{g_\mathrm{l}}^\mathrm{ctrl}$. The value of g_l cannot be measured directly and C_m is not exactly known for an individual neuron. Therefore, the precise relation between both cannot be determined. Still, normally it is sufficient to adjust the membranes time constant $\tau_\mathrm{m} = \frac{C_\mathrm{m}}{g_\mathrm{l}}$ to the desired value by iteratively tuning the value of $I_{g_\mathrm{l}}^\mathrm{ctrl}$ and measuring τ_m. See Section 5.2.3 for a detailed description of such a calibration method. $I_{g_\mathrm{l}}^\mathrm{ctrl}$ can be varied from 0.0 µA to 2.5 µA (HVD) with a 10 bit resolution. The resulting membrane time constants are in the range of approximately 50 ns to 150 ns (HTD).

Synaptic Conductance Courses The synaptic CCs are controlled by four voltages and two currents: The voltage ramps $V_{\mathrm{ramp}}(t)$ generated by the synapse drivers as described above in Section 2.1.2 have a base line which is not necessarily at 0 V, but which can have a programmable positive offset $V_{\mathrm{synbias}}^{\mathrm{ctrl}}$. Such an offset causes a permanent current generated by the synapse nodes and hence a permanent conductance between the connected membrane and the corresponding reversal potential. Bias conductances can be used to minimize damping effects caused by the parasitic conductances of the wiring between synapse drivers and membranes, because these are permanently pre-charged. A second voltage $V_{\mathrm{synstart}}^{\mathrm{ctrl}}$ determines the initial value from which the voltage ramp starts its rising phase. It can shorten the total rise time τ_{rise} significantly, a feature that is useful if a quantal conductance increase shall be approximated. Both $V_{\mathrm{synbias}}^{\mathrm{ctrl}}$ and $V_{\mathrm{synstart}}^{\mathrm{ctrl}}$ can be set from 0.0 V to 2.5 V with a 10-bit resolution.

The programmable current $I_{\tau_{\mathrm{rise}}}^{\mathrm{ctrl}}$ determines the slope of the rising voltage ramp and therewith the speed of the exponential increase of a synaptically generated, transient conductance. The parameter current $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ controls the amplitude of that ramp. Its value is mirrored in every synapse node that receives the voltage ramp and therewith guarantees a robustly controllable amplitude of the exponentially rising and falling current generated there. It generates a voltage that is permanently compared with $V_{\mathrm{ramp}}(t)$. If $V_{\mathrm{ramp}}(t)$ exceeds this voltage, the rise is flipped into a decay, the speed of which is determined by a third current $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$. The circuit that performs the comparison between $V_{\mathrm{ramp}}(t)$ and the voltage generated by $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ receives a configurable bias current $I_{\mathrm{syn}}^{\mathrm{bias}}$. If this current is set too low, the comparator becomes very slow, which might cause overshoots of $V_{\mathrm{ramp}}(t)$ over the value determined by $I_{\mathrm{amp}}^{\mathrm{ctrl}}$.

In terms of the model Equations 2.2 and 2.1, $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ determines $g_s^{\max}(t)$, while $I_{\tau_{\mathrm{rise}}}^{\mathrm{ctrl}}$ and $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ control $p_s(t)$. All four currents are programmable from 0.0 µA to 2.5 µA (`HVD`) with a 10 bit resolution.

Figure 2.5 shows a schematic of the circuitry that generates the described voltage ramp.

When adjusting $I_{\tau_{\mathrm{rise}}}^{\mathrm{ctrl}}$, there are constraints to be considered: Since many existing models that have been chosen for reference experiments use quantal or very fast increases of synaptic conductance as response to an incoming spike (see Section 1.3.2), τ_{rise} shall be as short as possible for the purposes of this thesis. But setting $I_{\tau_{\mathrm{rise}}}^{\mathrm{ctrl}}$ to values in the order of 1 µA or larger will cause an unwanted effect: The voltage ramp will significantly exceed the value determined by $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ due to a slow comparator mechanism. The speed of this comparator has been optimized by tuning the bias current $I_{\mathrm{syn}}^{\mathrm{bias}}$. For the `FHW-1.3` system, a value as small as $I_{\tau_{\mathrm{rise}}}^{\mathrm{ctrl}} = 0.2\,\mu\mathrm{A}$ has been found to deliver a sufficiently small rise time, since it is significantly smaller than the smallest achievable voltage ramp fall time (see hardware PSP[2] in Section 4.1.1).

Tuning $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ massively influences the efficacy of a synapse driver. But due to design-related issues described in Section 4.3.4, increasing $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ does not necessarily increase the CC amplitude monotonously. Still, if the values of $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ are kept limited, calibration mechanisms as the one presented in Section 5.2.4 can assume a monotonic relation.

Connectivity Every synaptic node can be individually programmed with a four-bit value which linearly scales the current kernels injected into the conductance control lines described above in Section 2.1.2. These four-bit values correspond to the synaptic weights $\omega_s(t)$ in Equation 2.2 and provide integer factors between 0 and 15.

[2]Post-Synaptic Potential

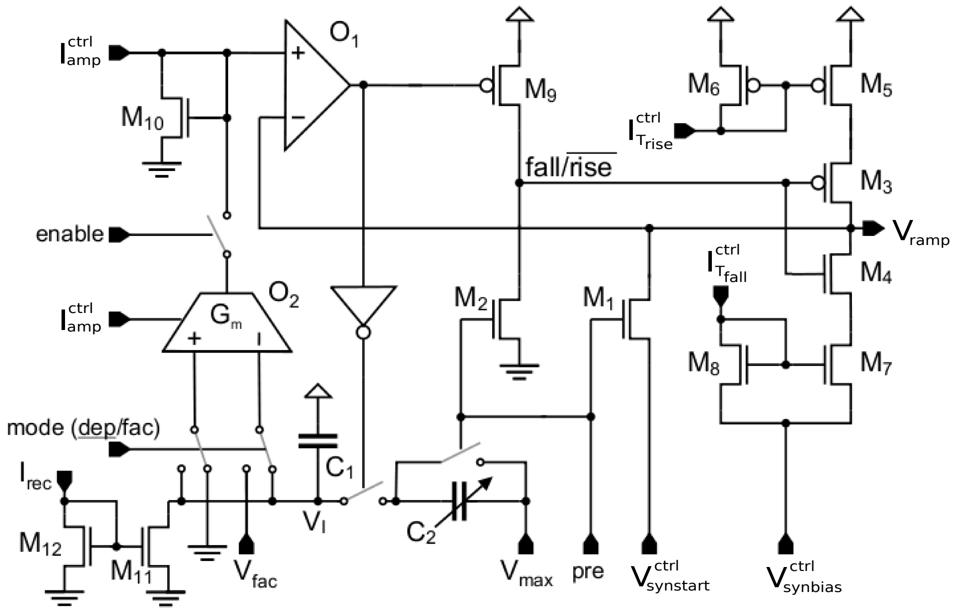


Figure 2.5: Diagram of the `FHW-1` synapse driver circuitry that generates the voltage ramp as indicated in Figure 2.3. This voltage ramp, which implements an intermediate state of generating the synapse response kernel, is represented by V_{ramp}. If a voltage pulse arrives at the `pre` pin, a linear increase of V_{ramp} via the current mirror M_5/M_6 fed by $I^{\mathrm{ctrl}}_{\tau_{\mathrm{rise}}}$ will be initiated, starting at value $V^{\mathrm{ctrl}}_{\mathrm{synstart}}$. Once a maximum voltage, defined by $I^{\mathrm{ctrl}}_{\mathrm{amp}}$ and M_{10}, is detected by the comparator O_1, the `fall/`$\overline{\texttt{rise}}$ line will be inverted and a decrease of V_{ramp} will be forced via the current mirror M_7/M_8 fed by $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}}$ until the resting value $V^{\mathrm{ctrl}}_{\mathrm{synbias}}$ is reached. Circuit diagram according to Schemmel et al., 2007, see there for more explanations.

STDP In order to find valid STDP configuration regimes for the hardware, multiple combinations of the parameters that control the amplitude of $F_{\mathrm{c}}(t)$, the amplitude of $F_{\mathrm{a}}(t)$ and the correlation threshold $V^{\mathrm{STDP}}_{\mathrm{thresh}}$ have to be swept.

In hardware, the amplitude of $F_{\mathrm{c}}(t)$, the amplitude of $F_{\mathrm{a}}(t)$ and the correlation threshold $V^{\mathrm{STDP}}_{\mathrm{thresh}}$ are controlled by four voltage parameters: $V^{\mathrm{clr}}_{\mathrm{a}}$, $V^{\mathrm{clr}}_{\mathrm{c}}$, $V^{\mathrm{ct}}_{\mathrm{high}}$ and $V^{\mathrm{ct}}_{\mathrm{low}}$. The difference $V^{\mathrm{ct}}_{\mathrm{high}} - V^{\mathrm{ct}}_{\mathrm{low}}$ controls the value of $V^{\mathrm{STDP}}_{\mathrm{thresh}}$, while $V^{\mathrm{clr}}_{\mathrm{a}}$ and $V^{\mathrm{clr}}_{\mathrm{c}}$ determine the amplitudes of $F_{\mathrm{a}}(\Delta t)$ and $F_{\mathrm{c}}(\Delta t)$. Like the neuron voltage parameters, they can be written in 10-bit values from 0 V to 2.5 V, but only values from approximately 0.6 V to 1.6 V are actually achievable. The `FHW-1` system does not provide a free parameter for adjusting the decay time constants of the modification curve. The available ranges of the amplitudes and the decay time constant values of the resulting STDP modification function have not yet been investigated systematically. Sections 4.1.3 and 6.1.5 show exemplary $F(\Delta t)$ curves acquired from hardware systems, though.

A so-called *synapse row* comprises all synapse nodes connected to the same synapse driver. The digital STDP controller can be configured to check and modify only specific synapse rows. Section 4.3.11 discusses possible problems arising from the cyclic update scheme, which can become rather slow if many synapse rows are involved.

Synaptic Depression and Facilitation The inactive partition described in Section 2.1.3 is electronically modeled by a voltage V_I. It can be seen in the lower left of the circuit schematic in Figure 2.5. If short-term synaptic plasticity is enabled (`enable` pin in diagram), the current $I_\mathrm{amp}^\mathrm{ctrl}$ will be changed by the depression and facilitation circuitry drawn in the lower left quarter of the schematic. Per synapse driver, only one option out of depression or facilitation can be chosen at a time by setting a corresponding `mode` bit. This choice influences the efficacy of all synapse nodes connected to this driver. The recovery time constant τ_rec is controlled via the current I_rec, which can be programmed by 10-bit values from $0.0\,\mu\mathrm{A}$ to $2.5\,\mu\mathrm{A}$. The voltages V_fac and V_max determine the values of λ_fac and λ_dep. Just like the neuron and the STDP voltage parameters, they can be written from $0\,\mathrm{V}$ to $2.5\,\mathrm{V}$ with a 10-bit resolution, but without specific workarounds (see Section 5.1.4), only the range from approximately $0.6\,\mathrm{V}$ to $1.6\,\mathrm{V}$ is available. This can cause serious problems if depression or facilitation are activated, see Section 4.3.10.

2.1.5 Stack of Hardware Communication Layers

Giving life to such a delicate piece of silicon like the `FHW-1` chip requires a whole set of support hardware devices, which, among others, provide mechanical footing, power supply, an infrastructure for the programmable control devices (FPGAs[3]), memory for the real-time playback of input stimuli and for the recording of output data, pins and amplifiers for analog observables, plus an interface for high-level control software modules running on external computers.

The following list summarizes the most important components of the support hardware hierarchy:

- Every `FHW-1` chip is mounted on a so-called *Recha* board (Ostendorf, 2007). These PCBs provide the baseplate for the bonding wires which fan out the chip pins to larger contacts, allowing to access it with normal PCB electronics. Furthermore, Recha boards have connectors via which the analog sub-threshold potentials of arbitrarily selectable neurons can be monitored. One further pin that is used in later sections of this thesis is a multiplexed pin, which can be programmed to output selectable membrane potentials or chip-internal parameter values via the programmable so-called I_b^{test} line. The chips are either packaged in cases or protected by plexiglass covers. In Figure 2.7, the board, an analog connector and the `FHW-1` chip under a plexiglass lid can be seen.

- Every Recha PCB sits on a so-called *Nathan* board (Grübl, 2007, Chapter 3; Fieres et al., 2004), which, among others, carries an FPGA for the digital controlling of the chip and – on its back side – a RAM[4] module for the storage of digital input and output data. Stimulation data can be delivered during an experiment run from this so-called *playback memory* in real time, using digital logic in the FPGA and on the chip itself. The same RAM module is used to intermediately store the events generated by the chip during an experiment, before it is read out by a controlling software. The bandwidth of this stimulus and output data transport between the chip and the RAM is limited, see Section 4.3.7. In Figure 2.7, a Nathan board can be seen, with the metal heat sink for the FPGA clearly visible.

[3]Field Programmable Gate Array
[4]Random Access Memory

- Every Nathan board is connected to a so-called *backplane* (Philipp et al., 2007). Up to 16 Nathan boards can be connected to one backplane at the same time and, via a dedicated network infrastructure, among each other. The backplane is connected to a power supply unit and distributes power to all Nathan boards. Figure 2.7 also shows the backplane.
- Via a custom-design transport medium based on SCSI[5] cables and connectors, the backplane is connected to a so-called *Darkwing* PCI[6] card (Schürmann et al., 2002) which is plugged into a *host PC*.
- That host PC is running a Linux operating system and performs communication with the backplane via a partly custom-design, partly proprietary Token-Ring-like (IEEE) network protocol (Fieres et al., 2004; Philipp, 2008). Currently, every software (see Section 3.2 and Brüderle et al., 2007) that is supposed to communicate with such an `FHW-1` chip has to use this protocol. A Gigabit Ethernet solution is under development, but was not available throughout the work for this thesis.

All digital information that can be read from the chip, e.g. the time stamps and neuron indices of any spikes fired during an experiment run, can be acquired via this communication stack. In order to access the analog sub-threshold membrane potential of selectable hardware neurons, an oscilloscope can be connected to dedicated Recha connector pins. One chip can output up to eight of its membrane potentials at the same time. The device[7] utilized for the work presented in this thesis provides the possibility to access the acquired digitized data via TCP/IP sockets (LeCroy, 2005; Braden, 1989). Dedicated modules of the higher-level software layers executed on the host PC can collect this information and integrate it into the scope of accessible observables (see Section 3.2.2).

Figure 2.6 summarizes the `FHW-1` support hardware setup schematically, while Figure 2.7 shows a photograph of the system as it actually exists in the laboratory.

[5]Small Computer System Interface
[6]Peripheral Component Interconnect
[7]LeCroy WaveRunner 44Xi

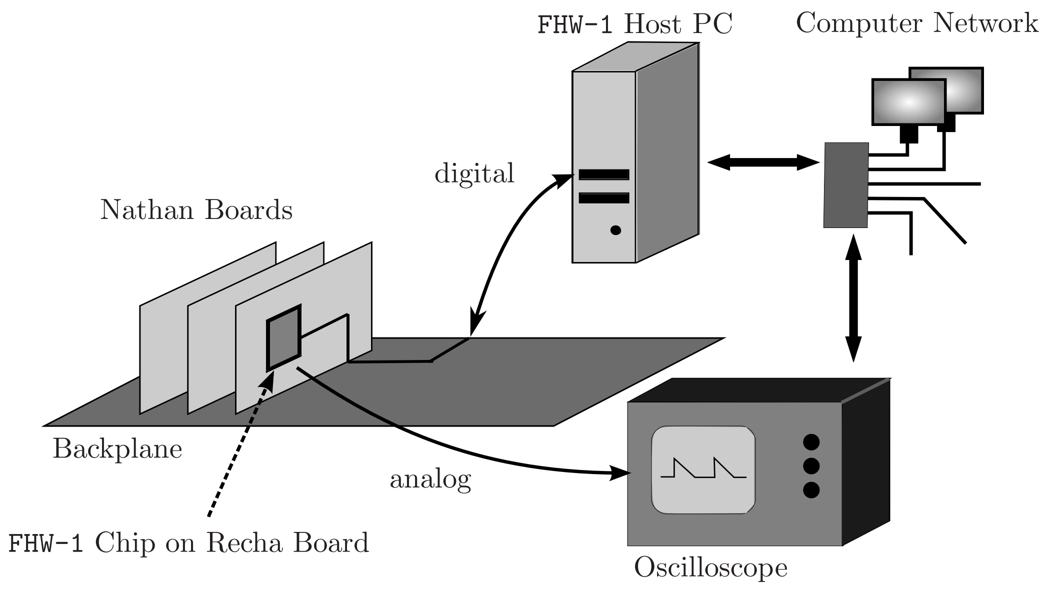


Figure 2.6: Schematic of the complete `FHW-1` hardware setup. The neural network chip is mounted on a Recha PCB, which itself sits on a Nathan board. Multiple Nathan boards are plugged into a backplane, which, among others, provides the power supply and which is digitally connected to a host PC. Connectors on the Recha boards allow to connect selectable analog sub-threshold membrane potentials on the `FHW-1` chip with an oscilloscope. The oscilloscope can be interfaced via a computer network and its acquired information about the analog membrane potentials can be integrated into a software scope (see Section 3.2.2) running on the host PC.

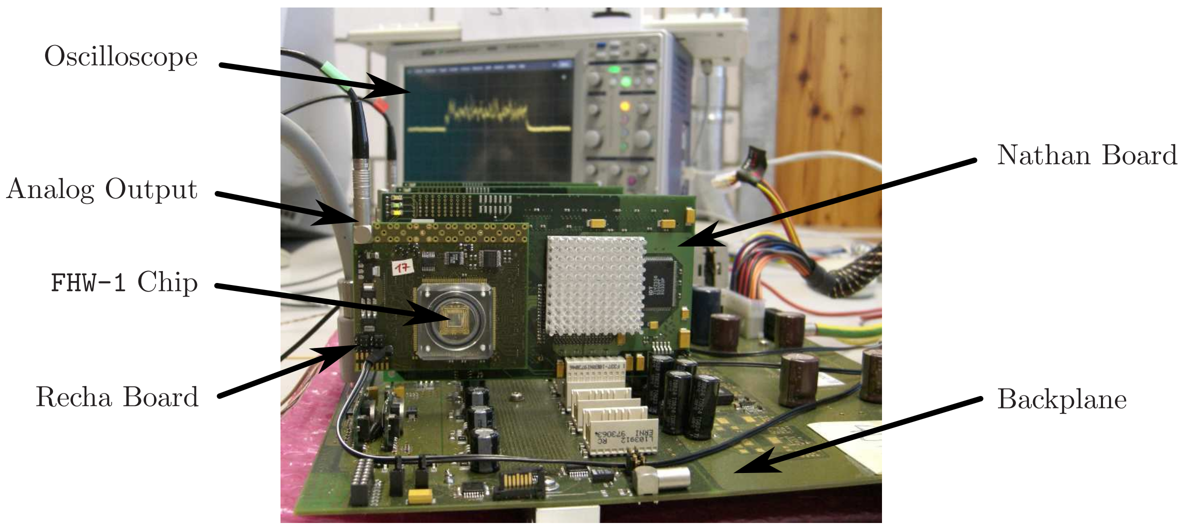


Figure 2.7: Photograph of a `FHW-1` system with support hardware. Compare with Figure 2.6.

2.2 Wafer-Scale Neural Network Model

One major purpose of the chip-based FACETS hardware system (`FHW-1`) introduced in Section 2.1 is to serve as a prototype for a much larger neuromorphic hardware device, the so-called *wafer-scale* model (`FHW-2`, Schemmel et al., 2008). Wafers are slices of semiconductor material, e.g. monocrystalline silicon, which are used in the fabrication process of micro-electronic devices. Into and upon the wafers, integrated circuits are built. Typical wafers are less than 1 mm thick and have a round shape with a diameter between 150 mm and 450 mm. In the process of industrial VLSI chip production (Hastings, 2001, Section 3.2.2), various steps are applied to the wafer in order to form the desired circuitry structure: E.g. doping of the pure semiconductor material with donors or acceptors, chemical etching and deposition of materials. For this purpose, sets of masks are applied photolithographically, each of which contains the spatial patterning information for a certain production step. These steps are applied sequentially, hence, one set of masks fully determines the vertical layer structure on the final device. But due to technical limitations, such a set can define only a fraction of the full wafer area (e.g. 20 mm $\times$ 20 mm), the so-called *reticle*. In a two-dimensional stepping process, this reticle is replicated multiple times on one wafer, i.e. the structure is identically repeated.

The smallest structures that can be produced in modern CMOS fabrication processes are in the order of hundreds and even tens of nanometers. On these microscopic scales of material manipulation, the occurrence of imperfections is unavoidable, e.g. by particles that distort the lithographic exposure process. Often, such production flaws destroy the functionality of the affected circuits. Since a certain rate of imperfection has to be expected for every wafer, the standard method to cope with this phenomenon is to produce chips with a limited size. In a typical chip production run, one reticle contains multiple instances of the same design, and after the wafer is fully produced, it is cut into pieces defined by the boundaries of these designs. Then, the functionality of each of the obtained chips is tested, and a certain fraction, the so-called *yield*, will turn out to be unaffected by production flaws.

Normally, digital designs rely on the correct operation of the underlying circuitry, i.e. one functionally relevant distortion in a chip is sufficient to make the whole device unusable. The same accounts for most analog designs, and the larger the area of a chip is, the larger grows the probability of a defect on each die. This extreme sensitivity of most micro-electronic circuits to the *full* functionality of the utilized substrate is intrinsic to the implemented architectures. In contrast to this, neural circuits, e.g. in mammalian cortices, exhibit an intrinsic fault tolerance. Their functionality is not significantly degraded by the loss of single neurons or connections, the occurrence of which is normal due to e.g. cell death or synaptic reorganization. This feature opens up new perspectives for the design of information processing architectures based on micro- or nanometer-scale technologies.

Given the possibility to transfer the said neural architectures to error-prone technologies like e.g. CMOS, the intrinsic fault tolerance of these circuits is expected to massively increase the yield in the production of the obtained devices. This concept is an essential prerequisite for the successful operation of the `FHW-2` system, which is currently under development within the FACETS research project (see Section 1.4). Uncut silicon wafers will serve as the substrate for these large-scale mixed-signal neuromorphic hardware devices. Implemented in CMOS technology, a large number of neuron and synapse circuits (see Section 2.2.1) will cover the full wafer while being highly configurable and interconnectable (see Section 2.2.2). For this purpose, the event communication infrastructure has to cross reticle borders. Such an interreticle wiring, which is not provided by commercial chip fabrication processes, will be applied in the form of a so-called *post-processing* metal layer on top of the wafer. In the following, the `FHW-2` system, the implemented neuron model, the number of realized cells and connections, the hardware dimensions and its mechanical design are described.

2.2.1 Technology and Dimensions

Like the `FHW-1` system described above in Section 2.1, the `FHW-2` system will be built using a standard 180 nm CMOS process. Every `FHW-2` unit incorporates one wafer, while multiple of such units will be interconnectable to set up larger networks. On each wafer, at least 44 reticles are used, and every reticle consists of 8 so-called HICANN[8] chips. Every chip implements more than 115000 synapses. The number of neurons per chip is configurable, values between 8 and 512 are possible. Depending on this number, the number of input synapses per neuron varies between 14848 (in case of 8 neurons per chip) and 232 (in case of 512 neurons per chip). In total, one wafer implements up to 180224 neurons and more than 40 million synapses. The size of one HICANN die is 50 mm^2, the utilized wafers have a diameter of 200 mm.

2.2.2 Implemented Model

The `FHW-2` system implements a so-called *adaptive exponential integrate-and-fire* (aEIF) neuron model with conductance-based synapses as described in Brette and Gerstner, 2005. Like in the `FHW-1` system, the cell bodies in the first version of the `FHW-2` system are modeled as *point neurons*, i.e. they do not comprise any compartments that are separated by resistors. A multi-compartment structure is planned to be realized in later versions. The system is designed to exhibit an acceleration factor of approximately 10^4.

Membrane Potential Dynamics

The neuron circuits are designed such that the temporal development of an emulated membrane potential $V(t)$ is determined by the following differential equation:

$$-C_\mathrm{m}\frac{dV}{dt} = g_\mathrm{l}(V-E_\mathrm{l}) - g_\mathrm{l}\,\Delta_\mathrm{th}\exp\left(\frac{V-V_\mathrm{th}}{\Delta_\mathrm{th}}\right) + g_\mathrm{e}(t)(V-E_\mathrm{e}) + g_\mathrm{i}(t)(V-E_\mathrm{i}) + w(t) \quad . \tag{2.7}$$

The variables C_m, g_l, E_l, E_e and E_i are defined like for the `FHW-1` membrane potentials (see Equation 2.1). The variables $g_\mathrm{e}(t)$ and $g_\mathrm{i}(t)$ represent the total excitatory and inhibitory synaptic conductances. The *exponential* term on the right hand side introduces a new mechanism to the I&F neuron: Under certain conditions, the membrane potential develops rapidly towards infinity. The *threshold potential* V_th represents the critical value above which this process can occur, and the *slope factor* Δ_th determines the rapidness of the triggered growth. Such a situation, which is detected by a mechanism that permanently compares $V(t)$ with a critical value $V_\mathrm{spike} > V_\mathrm{th}$, is interpreted as the emergence of a spike. Each time a spike is detected, a separately generated output event signal is transmitted to possibly connected target neurons (or to recording devices), and the membrane potential is forced to a reset potential V_reset by a very strong reset conductance.

[8]High Input Count Analog Neural Network

A second extension to the basic conductance-based I&F model described by Equation 2.1 is the so-called *adaptation current* $w(t)$. The value of w develops over time as follows:

$$-\tau_w \frac{dw}{dt} = w(t) - a(V - E_l) \qquad . \tag{2.8}$$

Additionally, every time a spike is emitted by the neuron, w changes its value quasi-instantaneously: $w \rightarrow w + b$. The time constant and the efficacy of the so-called *sub-threshold* adaptation mechanism are given by τ_w and a, while b defines the amount of the so-called *spike-triggered* adaptation.

For a detailed discussion of the possible neuron dynamics emerging from this model and for a fitting of the parameters to physiological data, see Brette and Gerstner, 2005. Since both the exponential term in Equation 2.7 and the adaptation can be deactivated, the `FHW-2` neurons can be configured to implement the same model as the `FHW-1` systems.

Connectivity and Synapse Model

Programmable Connections The event communication infrastructure in the `FHW-2` systems is organized into two hierarchical layers. So-called *Layer 1* (L1) buses transport spikes directly on the wafer. Local groups of up to 64 neighboring neurons on a HICANN chip feed one L1 bus line. A priority encoder manages the temporal multiplexing of their output signals onto the shared resource, where the events are transmitted in a serial, continuous time protocol. The L1 bus network consists of horizontal and vertical lines, which interconnect all HICANN chips on one wafer among each other. Repeaters at the boundaries of every HICANN relay the events and guarantee a sufficient signal quality even across long transportation distances. The wiring applied during the wafer post-processing provides the possibility to route the L1 lines across the reticle borders. At all crossing points of the horizontal and vertical lines, configurable so-called *crossbar switches* allow a flexible programming of the realized routing pattern. Once a spike signal has reached its destination HICANN, it can be fed into a synapse driver of that chip via programmable *select switches*. See Figure 2.8 for a schematic.

The `FHW-2` Layer 1 routing infrastructure allows for a flexible configurability of the on-wafer spike routing. In (Fieres et al., 2008) and (Schemmel et al., 2008), the L1 bus system is described in more detail. In (Fieres et al., 2008), algorithms are presented which determine configurations of the crossbar and select switches that realize given network topologies with a minimum of distortions.

The so-called *Layer 2* routing framework represents the possibility to record and digitalize an output spike of a neuron by off-wafer hardware modules (see Section 2.2.4), route them via a dedicated digital network to a target HICANN on the same or on another wafer system, and there feed them back into the analog Layer 1 bus system. This analog-to-digital conversion followed by the transport via a digital protocol followed by a digital-to-analog conversion is much more time consuming than a pure Layer 1 event transport. Typical hardware delays of a Layer 1 neuron-to-neuron spike transmission will correspond to less than 1 ms in `BTD`, while the Layer 2 delays can add up to approximately 5 ms (`BTD`).

Synapse Dynamics Like in the `FHW-1` system, conductance-based synapses are implemented, the dynamics of which are very similar to the alpha function shape or – if configured adequately – to the quantal increase followed by an exponential decay (see Section 1.3.2). Similar to the prototype implementation, the temporal shaping of conductance courses in the `FHW-2`

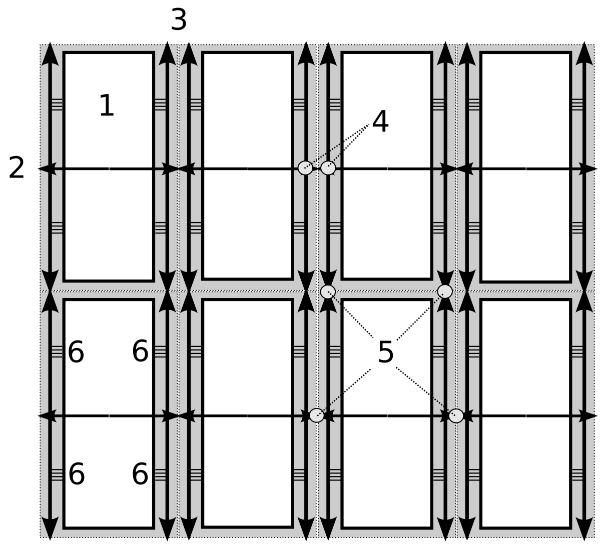


Figure 2.8: Schematic of the Layer 1 bus framework on the FHW-2 system, showing eight out of the 352 HICANN chips (1) per wafer. On each chip, one horizontal bus (2) with 64 lines and two vertical buses (3) with 128 lines each provide the transport of spike signals from one chip to another. Configurable crossbar switches (4) allow to route signals in both directions. Repeaters (5) provide the event transport across chip boundaries. The output spikes generated by the hardware neurons are fed into the horizontal bus structure that crosses every chip. After they have reached their destination chip, they can be selectively connected to synapse drivers on the target HICANN by programmable select switches (6).

system is determined by so-called synapse drivers. In synapse nodes, which represent a configurable switch matrix connecting the synapse drivers with the neurons on a chip, the conductance amplitude of a synaptic connection can be individually scaled by a four-bit value. See Section 2.1.2 for the corresponding prototype implementations.

Synaptic Plasticity The short-term synaptic plasticity mechanisms implemented in the FHW-2 systems will be basically the same like in the FHW-1 devices. For the long-term plasticity, a more flexible programmability of the weight modification functions (see Section 2.1.3 for the corresponding implementation in the FHW-1 system) is currently under development.

2.2.3 Configurability of the System

In contrast to the FHW-1 model, where many neurons share parameter values, nearly all parameters that determine the behavior of FHW-2 neurons will be configurable individually for every cell. For this purpose, so-called *floating gate* memory cells (Ehrlich et al., 2007) have been developed specifically for the application in the FHW-2 system. They allow to store continuous analog voltage values like e.g. the resting potential, the firing threshold or the reversal potentials of the hardware neurons in a very area- and power-efficient way.

Like in the chip-based prototype system, the synapse drivers in the FHW-2 devices will determine the temporal aspects of synaptic conductance courses. The amplitudes of synaptic conductances can be configured individually for every connection with a four-bit value. The realizable network topologies are determined by the possibilities provided by the Layer 1 and

Layer 2 routing framework as indicated above. See Section 3.1.6 and (Ehrlich et al., 2008; Wendt et al., 2008) for more information on the strategies that have been developed in order to map biological network architectures to the available hardware substrate with a minimum of distortions.

2.2.4 Stack of Hardware Communication Layers

The large number of neurons and synapses per `FHW-2` wafer, in combination with the speedup factor the system shall be operated with, implies highly challenging bandwidth requirements for the data transfer *to* and *from* the wafer. In order to fully configure the system, a large number of parameter values has to be transferred to the digital and analog memory cells. Externally generated stimulation data has to be injected into the Layer 2 event communication network, from where it is fed into the continuous time domain of the Layer 1 bus system. The output spikes of all recorded neurons have to be collected and transported off the wafer into the memory of a controlling host PC.

For the configuration, stimulation, controlling and recording of the neural substrate implemented in a `FHW-2` wafer, a complex support hardware framework has been developed. Figure 2.9 shows a schematic and summarizes the most important components. The physical signal exchange between the wafer and this framework is performed via elastomeric connectors that fill the space between the wafer and a large printed circuit board (PCB) on top of it. These connectors, which provide finely grained vertical conducting structures, establish the electrical contact between the post-processing metal layer on top of the wafer and dedicated pins on the PCB. Groups of eight HICANN chips are interfaced by one so-called DNC[9] chip each, which provides the event package generation and addressing for the Layer 2 routing. Groups of four DNC chips are interfaced and controlled by one FPGA each. These sub-units of four DNCs and one FPGA also incorporate memory devices, which store the Layer 2 routing information, externally generated stimulation data and the recorded output data generated on the wafer.

A total of twelve FPGAs are incorporated in one `FHW-2` system. Together with the DNCs, they perform the digital Layer 2 event routing and the distribution of all configuration data and control signals. Every FPGA also provides a Gigabit Ethernet communication channel to external controlling devices. Via these Gigabit connections, a host PC can send the configuration and stimulation data to the system and read back the generated output events.

The total worst case power consumption of this system is assumed to be approximately 2 kW, the half of which is caused by the wafer itself, the other half by the support hardware framework.

[9]Digital Network Chip

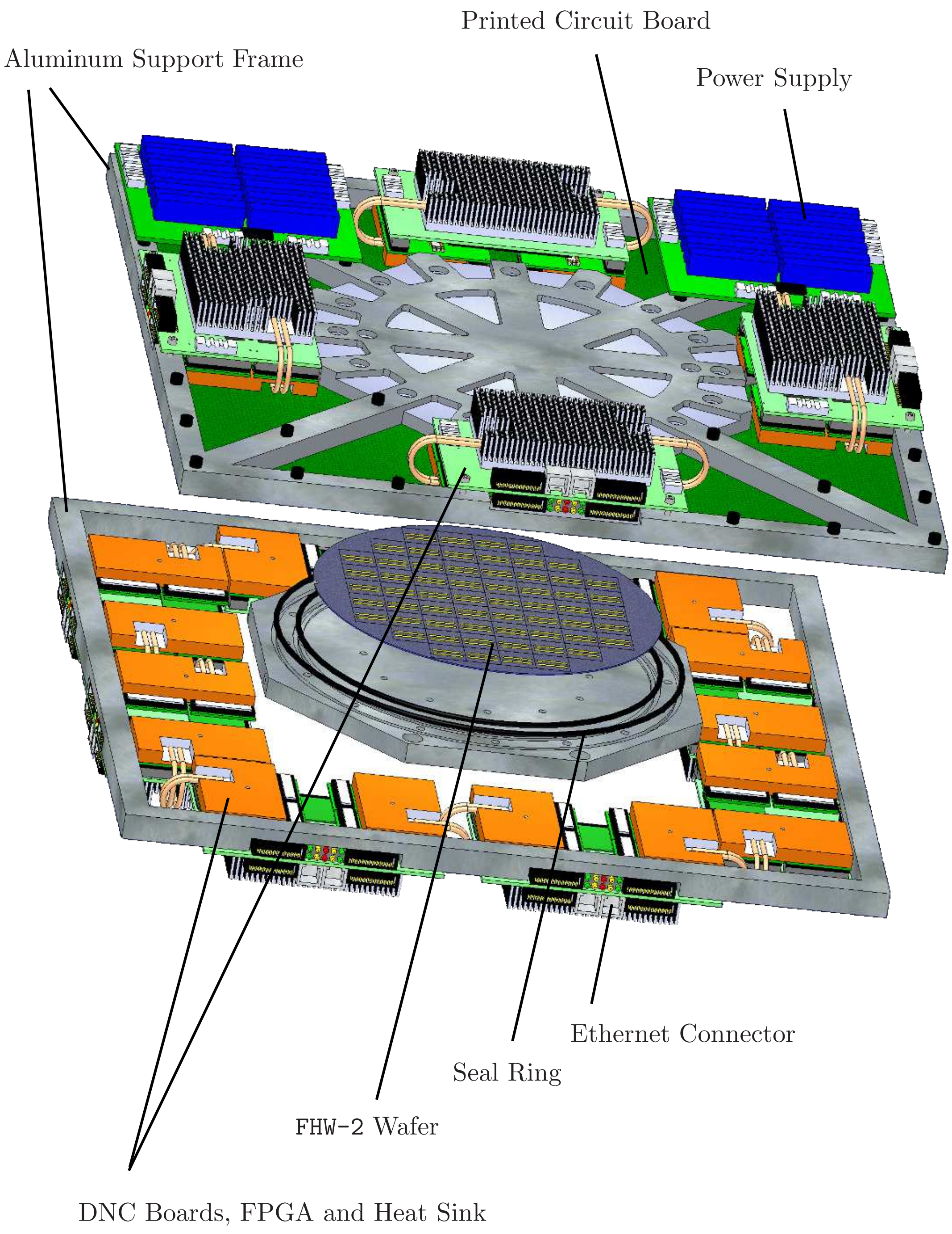


Figure 2.9: Schematic of one FHW-2 unit. The core of the system is a wafer that incorporates more than $180 \cdot 10^3$ neurons and more than $40 \cdot 10^6$ synapses. The wafer reticles are interconnected by post-processing wiring structures. Elastomeric connectors establish electrical contacts between pins on the surface of the wafer and a printed circuit board, which provides the routing of digitalized events between the wafer and the DNC chips. Those chips implement the exchange of event data between the Layer 2 and the Layer 1 routing networks. Via FPGAs, these DNC chips communicate among each other. The FPGAs also provide the interface between the Layer 2 network and a Gigabit Ethernet connection to external control devices. Figure by D. Husmann.

3 Software and Techniques

This chapter introduces concepts that have been developed in order to utilize the accelerated FACETS hardware systems (`FHW-1` and `FHW-2`) for neuro-scientific modeling. Section 3.1 describes paradigms of hardware operation that allow a biological interpretation of the output generated by such neuromorphic devices. The main addressed aspects are:

- An electric interpretation scheme, i.e. methods to translate between electric properties and observables of the hardware system and of the biological model, e.g. membrane potentials of neurons.
- A temporal interpretation scheme, i.e. methods to translate between hardware and biological time domain.
- A framework in which the translated hardware output can be compared with the output of established models, in this case with software simulators.

Section 3.2 describes a software architecture that has been implemented to realize the proposed paradigms. This software framework is *the* essential instrument for all hardware experiments presented in this thesis. Due to its underlying concepts and implemented methods, it is not only a technological framework, but provides a methodological fundament not only for the presented but also for future experiments with the `FHW-1` and `FHW-2` systems.

3.1 Operation Paradigms

One application of the `FHW-1` and `FHW-2` systems is to use them as modeling tools in computational neuroscience, complementary to the commonplace software simulators. In order to achieve this, concepts and methods have been developed which are introduced in the following.

Section 3.1.1 outlines typical utilization scenarios for `FHW-1` and `FHW-2` devices and discusses requirements for the user interface arising from such scenarios and from hardware-specific issues. Existing interfaces for hardware systems other than the FACETS type are shortly described in Section 3.1.2.

The back-end agnostic modeling language *PyNN* is introduced in Section 3.1.3, and the integration of the `FHW-1` and `FHW-2` interfaces into this framework is motivated and sketched. A short introduction on the two neural network software simulators *NEST* and *PCSIM* is given in Section 3.1.4, since they are used for comparison and reference applications within this thesis.

In Section 3.1.5, methods for the interpretation respectively translation of time scales and membrane potentials between the biological and the hardware domain are explained. Section 3.1.6 describes a technique to map the topology and the parameters of a biological network model to the limited resources of a neuromorphic device.

3.1.1 Intended Scenarios of Usage

When planning a user interface to a given device, its applications have to be considered. In the future, neuromorphic VLSI systems inspired by cortical circuits might provide the intelligence for autonomous machines, e.g. as the sensory-motor control and decision-making mechanism of robots. They might be embedded in standard computers as novel co-processors, finally turning them into truly intelligent and adaptive machines. Many visionary and complex applications of neuromorphic circuitry can be thought of, but before they become feasible, the understanding of cortical information processing has to develop significantly, and the technology itself has to be mastered.

The devices created within the FACETS research project, i.e. the `FHW-1` and the `FHW-2` systems, serve these two purposes: To learn about possible micro-electronic realizations of cortical architectures, and by doing so, to provide a tool for gaining more insights into cortical information processing by research on models.

Hence, intended scenarios of experimenting with the `FHW-1` and `FHW-2` systems are the following:

- Neural network models based on well defined connections between single cell units are realized by adequate configurations, stimulations and recordings of the hardware device (see Section 2.1.2, "What is an experiment?").
- A standard computer provides the physical interface for the user to administer the execution of experiments. Given an Internet connection and the appropriate software, this computer can be located everywhere, independently of the location of the operated hardware.
- Models are described in a digital, manipulable and storable format. Therein, parameters and observables are given in biological dimensions and follow a biological nomenclature. Like typical software simulators established in computational neuroscience[1], the hardware system provides an interpreter for the interactive or scripted setup and control of experiments.

[1]For example, NEURON (Hines and Carnevale, 2006; Hines et al., 2009) provides an interpreter called *Hoc*, NEST (Diesmann and Gewaltig, 2002; Gewaltig and Diesmann, 2007; Eppler et al., 2008) comes with a stack-based interface called SLI, and GENESIS (Bower and Beeman, 1998) has a different custom script language interpreter, also called SLI. Both NEURON and NEST also provide Python (Rossum, 2000) interfaces, as do the PCSIM (Pecevski et al., pending publication), Brian (Goodman and Brette, 2008) and MOOSE (Ray and Bhalla, 2008) simulators. An overview over existing software interfaces to hardware neural network models is given in Section 3.1.2.

- Experiment setups can be easily ported from at least one established software simulator to the hardware system and vice versa. The results from both back-ends can be compared in terms of a well-defined translation concept.
- Experiments can be performed by computational neuroscientists, who possibly have no or just a weak background in hardware engineering. Hence, the interface software must hide as many hardware-specific details as possible.
- The computation speed of the hardware devices is exploited by:
  - Acquisition of large statistics for every experiment. This is a weak point in many simulation experiments, and it is particularly important for analog hardware which does never perform two perfectly identical runs (see Section 4.2.2).
  - Searches for optimal configurations in large parameter spaces.
  - Emulation of long experiment durations. Especially due to the implemented synaptic plasticity principles (see Section 2.1.3), the FACETS hardware systems are predestined for studying long-term learning development.
  - Running experiments interactively. Since the on-chip execution time of typical experiments is extremely short on scales of human perception, intuition-guided manual parameter tuning with quasi-instantaneous feedback of results provides a novel way of experimenting and can help to explore and understand the possible dynamics of a setup (see Section 3.2.3).
- The chip-based `FHW-1` system is a prototype for the `FHW-2` wafer-scale system. While the latter will provide a substrate with sufficiently many neuron and synapse circuits to set up architectures relevant for topical neuroscientific research, the relatively small chip-based system used throughout this thesis fulfils other main purposes:
  - To test and specify those electronic circuits which will be used also in the large-scale system, i.e. to either prove their functionality or to revise and re-test the design until it works.
  - To test support hardware techniques (see Section 2.1.5).
  - To develop and test software concepts and to acquire operation experience.
  - To develop biological interpretation schemes and, within this scope, to explore the configurability of the neuron and synapse models.

These scenarios imply many requirements for the control interface, which are all considered by the actually implemented software framework (see Section 3.2).

3.1.2 Existing Hardware Interfaces

Descriptions in the literature of existing software interfaces to neuromorphic hardware systems are rare. In (Merolla and Boahen, 2006), the existence and main features of a GUI[2] for the interactive operation of a specific neuromorphic hardware device are mentioned.

Much more detailed software interface reports are found in (Dante et al., 2005). The authors describe a framework which allows the exchange of AER[3] data between hardware and software while experiments are running. The framework includes a dedicated PCI board

[2] Graphical User Interface

[3] Address Event Representation

which is connected to the neuromorphic hardware module and which can be interfaced to Linux systems by means of a device driver. A C-library layered on top of this driver is available. Using this, a client-server architecture has been implemented which allows the on-line operation of the hardware from within the program MATLAB. The use of MATLAB implies interpreter-based usage, scripting support, the possible integration of C and C++ code, optional graphical front-end programming and strong numerical support for data analysis. Nevertheless, the framework is somewhat stand-alone and does not facilitate the transfer of existing software models to the hardware.

In (Oster et al., 2005), an automatically generated graphical front-end for the manual tuning of hardware parameters is presented, including the convenient storing and loading of configurations. Originally, a similar approach was developed for the `FHW-1` system, too, see (Brüderle et al., 2007). Manually defining parts of the enormous parameter space provided by such a chip via sliders and check-boxes can be useful for intuition-guided hardware exploration and circuit testing, but it turns out to be rather impractical for setting up and controlling large network experiments as usually performed by computational neuroscientists.

3.1.3 Back-End Agnostic Description and Analysis

Multiple projects and initiatives provide databases and techniques for sharing or unifying neuroscientific modeling code, see for example the *NeuralEnsemble* initiative (Neural Ensemble, 2008), the databases of Yale's *SenseLab* (Hines et al., 2004) or the software database of the *International Neuroinformatics Coordination Facility* (INCF Software Database, 2008). Creating a bridge from the hardware interface to these pools of modeling experience will provide the important possibility of formulating transparent tests, benchmarks and requests that will boost further hardware development and its establishment as a modeling tool. The concept of integrating the `FHW-1` interface into the simulator-independent modeling language *PyNN*, which will be introduced below, shall implement such a inter-community bridge.

Choosing the Programming Language Python

The Integration Challenge As shown in Section 2.1.5, operating the `FHW-1` system involves multiple devices and mechanisms, e.g. a Message Queue communication with a user-space daemon that accesses a PCI board, TCP/IP socket connections to an oscilloscope, software modules that control the operation of the backplane, the Nathan board and the VLSI chip itself, and high-level software layers for experiment definition and flow control. On the software side, this multi-module system utilizes C, C++ and Python, and multiple developers from different institutions are involved in the development and maintenance of the code, applying various development styles such as object-oriented programming, reflective programming or sequential driver code. The software has to follow the ongoing system development, including changing and improving FPGA controller code and hardware revisions with new features.

This complexity and diversity argues strongly for a top-level software framework, which has to be capable of efficiently integrating all modules, supporting object-oriented and reflective structures, and providing the possibility of rapid prototyping in order to quickly adapt to technical developments at lower levels.

Python: An Efficient Glue Language With an interface to the `FHW-1` system which is based on the programming language Python (Rossum, 2000; Python, 2009), all scenarios listed in Section 3.1.1 are realizable. Python itself is an interpreter-based language with scripting support, thus it is able to provide a software-simulator-like interface. It can be efficiently connected to C and C++, for example via the package *Boost.Python* (Abrahams and Grosse-Kunstleve, 2003). This is important because the low-level software modules for direct communication with the support hardware framework (see Section 2.1.5), many of which have been developed and debugged over years, are written in C and C++. Re-implementing them would cause an unreasonably large effort and new possible error sources. Python supports sequential, object-oriented and reflective programming and it is widely praised for its rapid prototyping. Due to the possibility for modular code structure and embedded documentation, it has a high maintainability, which is essential in the context of a quickly evolving project with a high number of developers.

In addition to its strengths for controlling and interconnecting lower-level software layers, it can be used to write efficient post-processing tools for data analysis and visualization, since a wide range of available third-party packages offers a strong foundation for scientific computing (Oliphant, 2007; Jones et al., 2001; Langtangen, 2008), plotting (Hunter, 2007) and graphics (Summerfield, 2008; Lutz, 2001, chapter 8). Hence, a Python interface to the hardware system would already greatly facilitate modeler adoption.

Still, the possibility of directly transferring existing experiments to the hardware is even more desirable; a unified meta-language usable for both software simulators and the hardware could achieve that. Thus, the existence and growing acceptance of the Python-based, simulator-independent modeling language *PyNN* (see following section and PyNN, 2008; Davison et al., 2008) was the strongest argument for utilizing Python as a hardware interface, because the subsequent integration of this interface into PyNN depended on the possibility of accessing and controlling the hardware via Python (Brüderle et al., 2009).

Alternatives to Python Possible alternatives to Python as the top layer language for the hardware interface have been considered and dropped for different reasons. For example, C++ requires a good understanding of memory management, it has a complex syntax, and, compared to interpreted languages, has slower development cycles. The first GUI-based interface to the `FHW-1` system was pure C++ (Brüderle et al., 2007), but it turned out to be by far not as flexible as the Python approach followed since then, and implementing a similar scripting support would have cost significantly more effort. Interpreter-based languages such as Perl or Ruby also provide plotting functionality, numerical packages (Glazebrook and Economou, 1997; Berglihn, 2006) and techniques to wrap C/C++ code. Eventually, Python was preferred because it is considered to be easy to learn and to have a clean syntax, but especially because of the wish to be able to integrate this interface into the PyNN framework.

PyNN and NeuroTools

The advantages of Python as an interface and programming language are not limited to hardware back-ends. For the software simulators NEURON, NEST, PCSIM, MOOSE and Brian, Python interfaces exist. This provides the possibility of creating a Python-based, simulator-independent meta-language on top of all these back-ends. In the context of the FACETS project (see Section 1.4), the open-source Python module PyNN has been developed which implements such a unified front-end (PyNN, 2008; Davison et al., 2008).

Figure 3.1 illustrates the hierarchical structure of PyNN. For multiple simulation/emulation back-ends, it provides one unified access framework. If a neural network model is described in pure PyNN, the user only has to choose on which back-end it eventually will be executed. This is done by selecting one of the provided simulator-specific PyNN modules, which then automatically performs the translation of the PyNN model description into code that can be handled by the Python interpreter of the individual back-end. This Python interpreter actually operates the simulator kernel which is implicitly chosen with the PyNN module. Such a module selection is done with one line of PyNN code and allows to easily switch from one back-end to another.

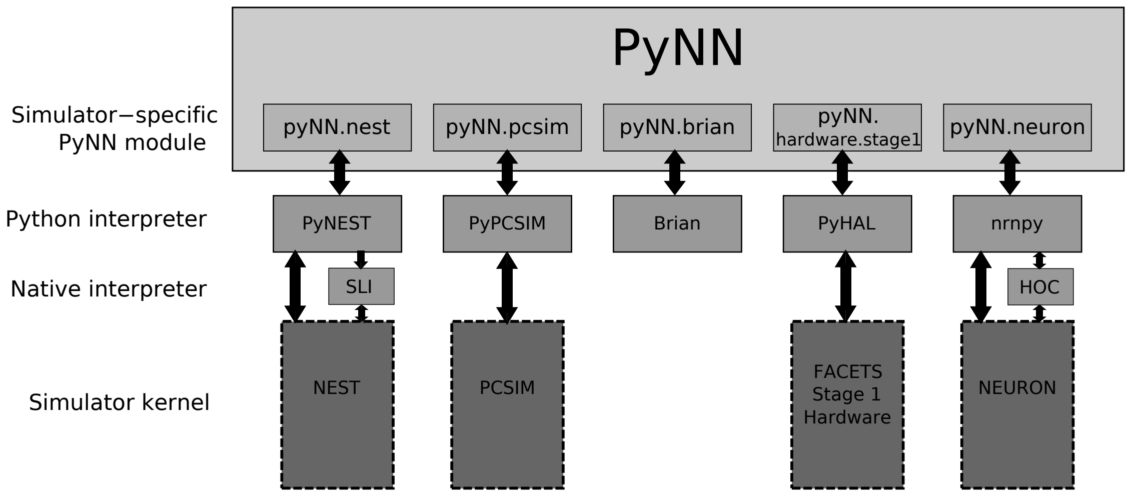


Figure 3.1: PyNN architecture: For the software simulators NEST, PCSIM, Brian, NEURON and for the `FHW-1` system, Python interpreters exist which communicate directly with the kernels or, as a further option for NEST and NEURON, via their native interpreters. PyNN is a unified interface to all these simulator- and emulator-specific Python interpreters. Back-end specific PyNN modules implement the translation to and from PyNN functions and classes to the individual lower-level software layers (black arrows denote direct communication).

Currently, PyNN supports the software simulators NEST, PCSIM, Brian and NEURON, plus the `FHW-1` system. Additionally, a module exists which generates NeuroML (Goddard et al., 2001) descriptions from PyNN code. The markup language NeuroML is a descriptive alternative to the sequential and/or object-oriented model unification approach of PyNN and can be interpreted, among others, by NEURON.

For the supported simulation back-ends, PyNN provides:

- A common interface.
  PyNN has a low-level procedural interface, i.e. a set of Python function declarations for the creation, manipulation, connection and operation of individual cells and synapses. For model descriptions on a higher abstraction level, PyNN also provides an object-oriented interface, i.e. classes like *Populations* and *Projections*, representing populations of neurons and connections between those. Only a fraction of the functionality of these functions and classes is actually implemented in PyNN. Most of it has to be realized for every individual underlying back-end.

- Built-in neuron and synapse models.

These models are standardized across the supported simulators. A certain PyNN built-in model represents the same set of differential equations on every supported back-end, and when given a set of parameter values, the corresponding translation to the individual back-ends will result in identical or, within the limits of numerical differences, very similar simulation output.

- Consistent handling of physical units.
  A typical problem which often complicates the transfer of a model from one simulation software to another is the different handling of units. For example, in the simulator NEURON, the value for a membrane capacitance is usually passed in nF, while in NEST the expected unit is pF. PyNN provides a clear set of units that are to be used for its various functions and cell types and performs the correct translation to and from the individual back-ends.
- Consistent handling of (pseudo-)random numbers.
  The generation of random numbers is an essential mechanisms in the realization of typical neural network models. External stimulation data has to be created according to certain distributions, e.g. Poisson-type spike trains, and the connectivity of the network is usually described by means of connection probabilities, i.e. the existence of every possible synapse has to be decided individually based on this information. If a certain experiment shall be repeated on different simulation or emulation back-ends, then the random number generation process has to be unified in order to have exactly the same stimuli to the network and in order to have an identical architecture. Usually, every back-end provides its own random number generation mechanisms, but PyNN provides the option to use the same, well-defined generator for all back-ends, resulting in the desired transparency and portability of random numbers.

With these features, PyNN offers the possibility of porting existing experiments between the supported software simulators and the `FHW-1` system and thus to benchmark and verify the hardware model. In Section 6.1.2, an example PyNN script is presented, which is executed on both the `FHW-1` and the NEST back-end.

On top of PyNN, a library of analysis tools called *NeuroTools* (NeuroTools, 2008) is under development, exploiting the possibility of a unified work flow within the scope of Python (see also Section 3.2.3). Experiment description, execution, result storage, analysis and plotting can be all done from within the PyNN and NeuroTools framework. Independent of the used back-end, all these steps have to be written only once and can then be run on each platform without further modifications.

Especially since the operation of the accelerated hardware generates large amounts of data at high iteration rates, a sophisticated analysis tool chain is necessary. For the experimental work presented in this thesis and for every possible PyNN user, making use of the unified analysis libraries based on the PyNN standards (e.g. NeuroTools) avoids redundant development and debugging efforts. This benefit is further enhanced by other third-party Python modules, like numerical or visualization packages (Oliphant, 2007; Numpy, 2008; Jones et al., 2001; Hunter, 2007).

PyNN for Hardware Calibration

The experiment portability inherent to the PyNN approach is the essential fundament upon which most of the calibration routines for the `FHW-1` system (see Section 5.2) have been developed. Based on PyNN, a software simulator which implements an appropriate model can be utilized for direct output comparisons even for complex setups. For two main reasons, such a reference mechanism can be extremely helpful when working with the `FHW-1` system: First, certain properties that need to be calibrated cannot be observed directly from the hardware, e.g. the synaptic conductance courses (see Section 2.1.2). Hence, these properties need to be derived using indirect access methods (see Section 4.1), and for those, the desired model output is not always trivial to determine. A corresponding simulation based on the PyNN script – which then has been written for the hardware anyway – provides reliable target values. And second, a calibration scheme which isolatedly steps from circuit to circuit might neglect existing interferences and activity dependencies intrinsic to the hardware model (see Section 4.3). Therefore, in order to increase the validity of a calibration, it should be performed with a setup as similar to the planned experiment the chip is calibrated for as possible. This increases the problem of predicting the correct output even more, and reference software simulations based on the same PyNN descriptions become an indispensable tool.

3.1.4 Reference Software Simulators

In different sections of this thesis (e.g. in Section 5.2.4, 6.2.1 and 6.2.2), the following two software simulators are used for the kind of reference experiments indicated in Section 3.1.3.

Neural Network Simulator NEST

The software simulator NEST[4] (Diesmann and Gewaltig, 2002; Gewaltig and Diesmann, 2007; Eppler et al., 2008; The NEST Initiative, 2009) is a framework for simulating large networks of biologically realistic neurons. It provides various synapse types, recording devices and neuron models and can be extended by user-written modules.

The neuron model utilized in all NEST simulations presented throughout this thesis is described in detail in Muller (2006). It exactly implements the conductance-based leaky I&F point neuron model as approximated by the `FHW-1` circuitry (see Equations 2.1 and 2.2), with conductance course kernels shaped as quantal increases followed by an exponential decay (see Equation 1.4). The optional mechanisms of spike frequency adaptation and relative refractoriness have been disabled, because they are not supported by the hardware.

NEST can be interfaced through the simulator-independent scripting language PyNN (see Section 3.1.3 and Davison et al., 2008; PyNN, 2008), allowing for a unified description and analysis of all experiments performed both in NEST and with the hardware (Brüderle et al., 2007).

Neural Network Simulator PCSIM

The software simulator PCSIM[5] (Pecevski et al., pending publication) is *"a tool for distributed simulation of heterogeneous networks composed of different model neurons and synapses"* (Pecevski and Natschläger, 2008). Like NEST, it provides modules which simulate exactly the model approximated by the hardware, including the exponentially decaying synaptic conductance courses (see Equations 2.1, 2.2 and 1.4). For the experiments presented in

[4]NEural Simulation Technology
[5]Parallel neural Circuit SIMulator

Section 6.2.2, this software simulator has been utilized, because within its framework a short-term synaptic plasticity module has been implemented by Johannes Bill and Klaus Schuch (Bill, 2008) which mimics the optional hardware behavior of synaptic depression and facilitation (see Section 2.1.3). Just like NEST, PCSIM can be interfaced through PyNN (see Section 3.1.3 and Davison et al., 2008; PyNN, 2008).

3.1.5 Neuron and Synapse Model Mapping

In Section 2.1.2, the differential equations are summarized which describe the hardware neuron and synapse model. In Section 2.1.4, the hardware parameters that actually have to be written to a `FHW-1` device are listed, including the hardware voltages which correspond to the biological neuron parameters. In this section, a set of mechanisms is explained that has been developed in order to provide a biological interpretation of the measured hardware variables, and to control them with parameters that are given in a purely biological nomenclature. Therefore, the deployed neuron voltage parameter translations, synaptic weight translations and time transformations are explained in the following. The mapping of network topologies is described in Section 3.1.6.

Neuron Parameter Mapping

The possibility to give a biological interpretation to the developing voltages of a hardware neuron cell is *the* essential feature of analog or mixed-signal neuromorphic hardware. Nevertheless, the applied translation procedure is not necessarily trivial.

Programmable Voltage Generators The neuron model parameters E_l, E_e and E_i as well as V_reset and V_thresh are programmable voltages generated on-chip by dedicated circuits (see Section 2.1.4, "Neuron Voltage Parameters"). For a certain range of input values U_in to these circuits, the generated voltages U_out depend linearly on the written value, while outside of this range the output voltage saturates on constant values:

$$U_\mathrm{out}(U_\mathrm{in}) = \begin{cases} U_\mathrm{out}^\mathrm{min}\,, & \text{if } U_\mathrm{in} < U_\mathrm{lower} \\ U_\mathrm{off} + m_\mathrm{u} \cdot U_\mathrm{in}\,, & \text{if } U_\mathrm{lower} \leq U_\mathrm{in} \leq U_\mathrm{upper} \\ U_\mathrm{out}^\mathrm{max}\,, & \text{if } U_\mathrm{upper} < U_\mathrm{in} \end{cases} \quad . \tag{3.1}$$

$U_\mathrm{out}^\mathrm{min}$ and $U_\mathrm{out}^\mathrm{max}$ limit the linearly programmable range, and U_off and m_u are the offset and the slope of the linear dependency. They are determined by a calibration routine, see Section 5.2.1. While for U_in values from 0 V to 2.5 V (`HVD`) can be written, typical values for the output minimum and maximum lie around $U_\mathrm{out}^\mathrm{min} \approx 0.6\,\mathrm{V}$ and $U_\mathrm{out}^\mathrm{max} \approx 1.6\,\mathrm{V}$ (`HVD`). For the `FHW-1.3-No.18` chip, for example, the mean and the standard deviation of $U_\mathrm{out}^\mathrm{min}$ over all voltage generators as found by the calibration routine is $\langle U_\mathrm{out}^\mathrm{min} \rangle \pm \sigma(U_\mathrm{out}^\mathrm{min}) = (0.57 \pm 0.02)\,\mathrm{V}$ (`HVD`). For the same chip, the routine found $\langle U_\mathrm{out}^\mathrm{max} \rangle \pm \sigma(U_\mathrm{out}^\mathrm{max}) = (1.625 \pm 0.002)\,\mathrm{V}$ (`HVD`), $\langle m_\mathrm{u} \rangle \pm \sigma(m_\mathrm{u}) = 0.948 \pm 0.005$, and $\langle U_\mathrm{off} \rangle \pm \sigma(U_\mathrm{off}) = (0.08 \pm 0.02)\,\mathrm{V}$ (`HVD`).

Assumptions For standard experiment configurations, the following assumptions are made in order to achieve a hardware-to-biology voltage mapping which exploits as much of the available dynamic range as possible:

- The neuron voltage parameters with the smallest values are either the reset potential V_reset or the inhibitory reversal potential E_i.

- The neuron voltage parameters with the largest value is always the excitatory reversal potential E_e.
- The firing threshold V_thresh has a value between those two extrema.
- The neuron resting potential has a value between V_thresh the maximum of V_reset and E_i.

These assumptions are based on typical parameter values used in cortical models, e.g. in (Kumar et al., 2008; Brette and Gerstner, 2005; Sussillo et al., 2007; Shelley et al., 2002). These have led to the default neuron voltage parameter set applied to most experiments presented in this thesis (see Appendix A.1): $V_\mathrm{reset} = E_\mathrm{i} = -80\,\mathrm{mV}$, $V_\mathrm{rest} = -75\,\mathrm{mV}$, $V_\mathrm{thresh} = -55\,\mathrm{mV}$ and $E_\mathrm{e} = 0\,\mathrm{mV}$ (`BVD`).

Finding the Lowest Possible Neuron Voltages For a neuron configuration according to the model defined in Equation 2.1 with a biologically relevant configuration, there are two candidates for the lowest voltage parameter: V_reset and E_i. For every `FHW-1` chip, a total of four values per voltage parameter exist: All neurons on the left network block with even indices share one value, those with odd indices share another, and the same accounts for the right network block. Correspondingly, there exist four minimum output values for the voltage generators of V_reset and four minimum output values for the voltage generators of E_i. The maximum of these eight minimum output values is considered as the minimum voltage V_lower a neuron on this chip can access.

Finding the Highest Possible Neuron Voltages There is only one candidates for the highest voltage parameter: E_e. Again, for every `FHW-1` chip four values exist for this parameter. Correspondingly, there exist four maximum output values for the voltage generators of E_e. The minimum of these four maximum output values is considered as the maximum reference voltage V_upper a neuron on this chip can have.

There is one further constraint to be considered: The range within which the emulated membrane potential $V(t)$ can evolve has a design-inherent upper limit of $V_\mathrm{max} \approx 1.2\,\mathrm{V}$ (`HVD`, Schemmel, 2008). Since the value of this limit can vary from chip to chip and from neuron to neuron, a strict upper limit for the firing threshold of $V_\mathrm{thresh}^\mathrm{max} = 1.1\,\mathrm{V}$ (`HVD`) is defined.

Determining the Final Voltage Mapping One requirement is strictly claimed: The translation of the membrane potentials emulated by the hardware neuron circuits into their biological interpretation has to be linear – this is what the hardware model has been designed for. Still, this does not necessarily mean that all reference voltages (reversal potentials, firing threshold, reset potential) have to be arranged with a linear correspondence to the biological model either.

Two voltage ranges determine the possible membrane potential states of a hardware neuron: The limits for the programmable voltage parameters of a neuron, V_upper and V_lower, span a programmable range ΔV_prog. The voltage range ΔV_mem which is actually accessible by a membrane potential is limited by $V_\mathrm{thresh}^\mathrm{max}$ and V_lower. Figure 3.2 illustrates these two nested regions.

For typical values in biological models, the ratio between the super-threshold and the sub-threshold voltage range

$$R_\mathrm{sub}^\mathrm{super} \equiv \frac{E_\mathrm{e} - V_\mathrm{thresh}}{V_\mathrm{thresh} - E_\mathrm{i}} \tag{3.2}$$

is larger than 2/1 (see Figure 3.2). In hardware, though, the corresponding ratio is less than 1/2 due to the limited ranges ΔV_{prog} and ΔV_{mem}. This discrepancy has been technically solved by implementing very strong synaptically induced conductances towards the excitatory reversal potential, resulting in realistic driving forces towards values above the firing threshold. And since the firing mechanism interrupts any evolution of the membrane potential above the firing threshold, the resulting membrane behavior is a good approximation of a biologically realistic situation with a larger value of E_{e}, but with lower synaptic conductances.

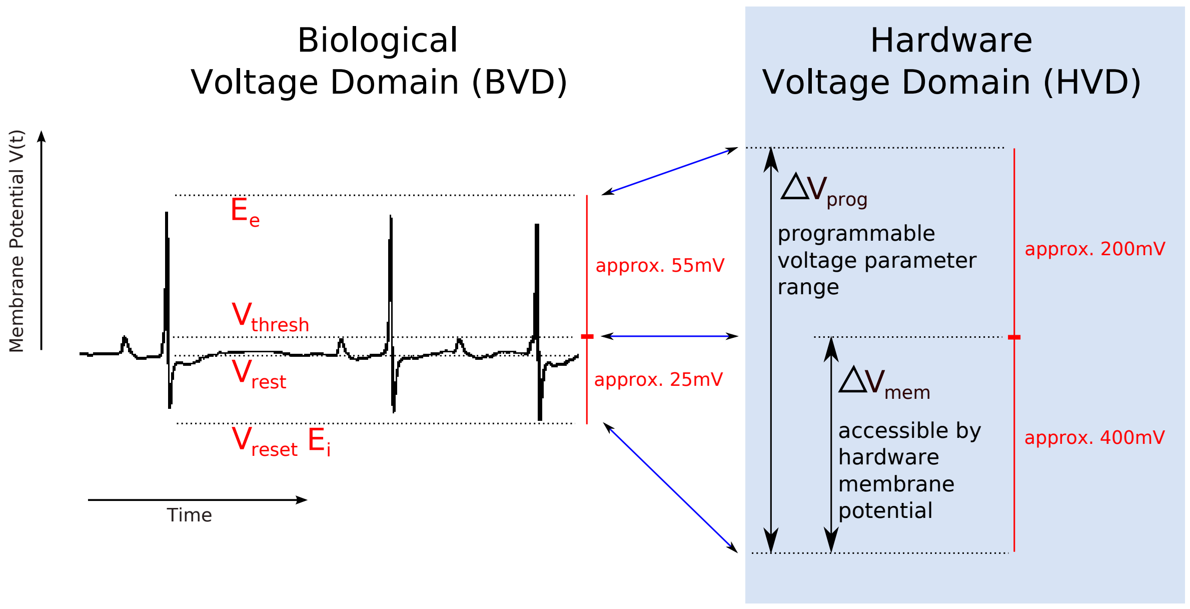


Figure 3.2: Schematic of the mapping between biological and hardware voltages. The voltage parameters of an evolving neuron membrane potential $V(t)$ (black trace) are E_{i}, V_{reset}, V_{rest}, V_{thresh} and E_{e} (see Equation 2.1). The hardware does not emulate the super-threshold dynamics of an action potential (see Section 2.1.2). The sub-threshold voltage range of the biological domain is linearly mapped to the voltage range ΔV_{mem} which is accessible by a hardware membrane potential. The super-threshold parameter E_{e} is handled differently, because the programmable voltage parameter range ΔV_{prog} in hardware does not provide values that are large enough to continue the linear sub-threshold mapping. See text for the corresponding technical solution.

Considering all constraints and hardware-specific issues, the final voltage mapping is determined as follows:

- Choose a ratio $R_{\mathrm{sub}}^{\mathrm{super}}$ for the hardware. It has to be kept fixed for all following calibrations and experiments in order to provide reproducible efficacies for inhibitory and excitatory synapses.
- Check if $E_{\mathrm{e}}^{\mathrm{test}} \equiv V_{\mathrm{thresh}}^{\mathrm{max}} + R_{\mathrm{sub}}^{\mathrm{super}} \cdot (V_{\mathrm{thresh}}^{\mathrm{max}} - V_{\mathrm{lower}}) < V_{\mathrm{upper}}$.
  - If yes, set $V_{\mathrm{thresh}} = V_{\mathrm{thresh}}^{\mathrm{max}}$ and $E_{\mathrm{e}} = E_{\mathrm{e}}^{\mathrm{test}}$.
  - If no, set $V_{\mathrm{thresh}} = V_{\mathrm{lower}} + (V_{\mathrm{upper}} - V_{\mathrm{lower}})/(R_{\mathrm{sub}}^{\mathrm{super}} + 1)$ and $E_{\mathrm{e}} = V_{\mathrm{upper}}$, i.e. decrease E_{e} to the largest available value and decrease V_{thresh} such that the ratio $R_{\mathrm{sub}}^{\mathrm{super}}$ is met.

This procedure guarantees the preservation of $R_{\text{sub}}^{\text{super}}$ while not violating the constraints imposed by $V_{\text{thresh}}^{\text{max}}$ and by V_{lower} and V_{upper}. For all experiments and calibration runs on `FHW-1.3` system presented in this thesis, the described method with a value of $R_{\text{sub}}^{\text{super}} = 2/3$ has been utilized. The fine-tuning of inhibitory and excitatory synapse efficacies still has to be performed by an adequate calibration routine (see Section 5.2.4), but the chosen mapping provides a starting point from which such a routine functions well.

Mapping of Synaptic Weights

As explained in Section 2.1.2, the synaptic signal transmission in the `FHW-1` system is comprised of three stages:

- The output of a pre-synaptic neuron or an external spike source connects to a so-called *synapse driver*. Every time such a driver receives a digital spike, it generates a linearly rising and then linearly falling voltage ramp.
- Every synapse driver connects to so-called *synapse nodes*, where the generated voltage ramps are transformed into currents. The amplitudes of these exponentially rising and exponentially falling current courses depend on the amplitude of the voltage ramp, but also linearly on the weight stored as a four-bit value in every synapse node. The time constants of the current courses are determined only by the durations of the linear voltage ramps.
- These currents are routed to post-synaptic neuron circuits, where they control the conductance between the neuron membrane and a reversal potential. Every neuron has two input current lines, one for a conductance towards its excitatory reversal potential, and one for a conductance towards its inhibitory reversal potential. Each line receives and sums up the currents from many synapse nodes.

With the rising part of the ramp being configured to be as short as possible, the resulting synaptic response kernel in the `FHW-1` systems can be described by a quasi-instantaneous increase of conductance followed by an exponential decay back to zero. Hence, in typical biological models which utilize the same kernel (see e.g. Sussillo et al., 2007 and Maass et al., 2004a), for every synapse the amplitude and the time constant of the exponential decay are the only two parameters given. In the context of such models, this kernel amplitude is often referred to as the *synaptic weight*.

Equation 2.2 reflects the hardware-specific separation of the synaptic conductance course amplitude into two factors:

$$g^{\text{amp}}(t) = \omega(t)\, g^{\text{max}}(t) \qquad , \tag{3.3}$$

where $g^{\text{max}}(t)$ is the contribution of the voltage ramp generated in the synapse drivers, and $\omega(t)$ is the four-bit weight in the synapse nodes.

The synapse driver contribution is utilized to provide a basic biology-to-hardware translation factor $T^{\text{bio-hw}}$. A method to determine the very device-specific value of $T^{\text{bio-hw}}$ is introduced in Section 5.2.4. But since every synapse driver serves many synapse nodes, this is only a global pre-transformation between the two domains. The heterogeneity of the synaptic weight values g_s^{bio} in a biological model is mainly reflected by individually different values of ω_s:

$$\omega_s = C\left(g_s^{\text{bio}} \cdot T^{\text{bio-hw}}\right) \quad , \text{with } \omega_s \in \{0, 1, ..., 15\} \qquad , \tag{3.4}$$

where C is a function that performs the discretization and the limiting of the product $g_s^{\text{bio}} \cdot T^{\text{bio-hw}}$ necessary due to the four-bit nature of the hardware weights. The following technique is applied in order to minimize the distortions caused by C, i.e. to provide a maximally fair mapping from the biological to the hardware weight space:

For every pre-translation $\widetilde{\omega}_s \equiv g_s^{\text{bio}} \cdot T^{\text{bio-hw}}$, a new random number r is chosen from a uniform distribution between 0 and 1. Then ω_s is determined as follows:

$$\omega_s = \begin{cases} \lceil \widetilde{\omega}_s \rceil & \text{if } r < (\widetilde{\omega}_s - \lfloor \widetilde{\omega}_s \rfloor) \\ \lfloor \widetilde{\omega}_s \rfloor & \text{else} \end{cases} \quad . \tag{3.5}$$

The Gaussian brackets $\lceil\ \rceil$ and $\lfloor\ \rfloor$ indicate that the embraced argument is rounded up respectively down.

This method introduces a random feature which, for a large number of synapse mappings, provides the correct average synaptic weights like in the biological original.

Mapping of Timescales

The hardware is designed to emulate the membrane potentials of cortical neurons with a certain speedup factor. This factor is determined by two time constants that are intrinsic to the hardware circuits and by the desired corresponding values in the biological model: The membrane time constant $\tau_{\text{m}} = \frac{C_{\text{m}}}{g_{\text{l}}}$ and the decay time constants of the synaptic conductance courses (see Section 2.1.2).

Typical membrane time constants for conductance-based I&F neuron models are found between 10 ms and 20 ms (`BTD`) (Kumar et al., 2008; Brette and Gerstner, 2005; Sussillo et al., 2007; Shelley et al., 2002). Synaptic decay time constants usually cover a much wider range, with typical values for excitatory synaptic conductance decays $\tau_{\text{syn,E}}$ chosen between 0.3 ms (Kumar et al., 2008) and 4 ms (`BTD`) (Sussillo et al., 2007; Shelley et al., 2002), while inhibitory synapses usually are configured to be a factor 2 to 4 slower.

Hence, an ideal hardware system would provide a membrane time constant which was adjustable within some range $[\tau_{\text{m}}^{\text{min}}, \tau_{\text{m}}^{\text{max}}]$, with $\tau_{\text{m}}^{\text{max}} \geq 2\,\tau_{\text{m}}^{\text{min}}$. In such a system, the synaptic decay time constants should then be adjustable within $[\tau_{\text{syn}}^{\text{min}}, \tau_{\text{syn}}^{\text{max}}]$, with $\tau_{\text{syn}}^{\text{min}} \leq 0.03\,\tau_{\text{m}}^{\text{min}}$ and $\tau_{\text{syn}}^{\text{max}} \geq 2\,\tau_{\text{m}}^{\text{min}}$.

For the `FHW-1` chips, $[\tau_{\text{m}}^{\text{min}}, \tau_{\text{m}}^{\text{max}}]$ has been found to be approximately [50 ns, 150 ns] (`HTD`, see Section 4.3.5). This suggests a `HTD`-to-`BTD` translation factor of $1 \cdot 10^5$ to $2 \cdot 10^5$. But the accessible synaptic time constants in the `FHW-1` systems are in the range of approximately [0.3 µs, 1.0 µs] (`HTD`, see Section 4.3.5). This is not in accordance with the requirements claimed above – a range of approximately [2 ns, 100 ns] would have been appropriate. In Section 4.3.5, this problem is discussed, and the finally chosen speedup factor of 10^5 is motivated.

3.1.6 Network Topology Mapping

A major constraint of neuromorphic hardware systems compared to software simulators is the strictly limited number of neurons and synapses that can be modeled. There is a well specified number of circuits per device which can emulate neural behavior. Hence, for a given abstract description of a biological network model, e.g. in form of a PyNN script (see Section 3.1.3), mapping this network onto the available number of neural circuits and the available connectivity circuitry is a crucial step within the full process of giving a biological interpretation to the hardware dynamics.

Designing large-scale neuromorphic hardware is always a trade-off between providing connection flexibility and achieving a high expected resource exploitation. Many assumptions regarding the kind of models to be emulated by the hardware have to be considered, and one important reason for the complexity of this connectivity challenge is the discrepancy between the 3D nature of cortical architectures and the basically 2D nature of CMOS technology. It is clear that for any reasonably chosen balance between flexibility and resource efficiency of a hardware solution, networks can be found that cannot be mapped to the provided structures perfectly.

But if a certain functionality inherent to a given cortical architecture is assumed to tolerate the loss of individual connections or even cells, as it usually is the case for any kind of *population coding* (Georgopoulos et al., 1986; Jaeger et al., 2007; Maass et al., 2002; Yamazaki and Tanaka, 2007), then many such network architectures can be mapped to the limited resources of a hardware device, e.g. within a given synapse loss tolerance.

Mapping in the Chip-Based System For the `FHW-1` system, which serves as a prototype for the much larger wafer-scale `FHW-2` system currently under development, the available neural substrate is in the order of a few hundred neurons per chip (see Section 2.1.1), and its stimulation and feedback infrastructure is rather simple, at least compared to the multi-level signal routing scheme of the `FHW-2` system (see Section 2.2). Thus, mapping a given biological network to an `FHW-1` chip with a minimum of topological distortions is something that can still be mastered by a human user. Still, the software framework for this system (which will be described in Section 3.2) provides some basic mapping features:

- The neurons of a biological network description are mapped to the available hardware neuron circuits one by one, i.e. the chip resources are filled up sequentially.
- A selectable mapping offset can be applied to this procedure, i.e. the filling starts at a configurable hardware index.
- Neurons which have been determined to be unusable by means of a calibration criterion (see Section 5.2.2) can be excluded from the set of available hardware neurons, i.e. the filling procedure will skip those circuits.

Mapping in the Wafer-Scale System For the `FHW-2` system, though, the pure number of neurons and synapses, the different possible levels of inter-neuron connections and the inter-dependencies of resource allocations and of parameter configurations make a much more sophisticated software mapping tool inevitable.

As a part of the `FHW-2` operating framework, Karsten Wendt from the Technical University of Dresden has developed a topology and parameter mapping strategy based on the representation of both the biological model and the hardware substrate as graphs (Wendt et al., 2008; Ehrlich et al., 2008). This so-called *GraphModel* mapping approach is schematically illustrated in Figure 3.3. The abstract biological model, e.g. originally described in a PyNN script, is converted into a hierarchical graph structure. A root node represents the biological model as a whole and has various child nodes connected via so-called *hierarchical edges*, such as a neuron top-level node, a neuron parameter top-level node and a synapse parameter top-level node.

The children of the neuron top-level node are all neurons within the biological model, again connected to their parent node via hierarchical edges. Synaptic connections between these neurons are represented by *synapse edges* between the neuron nodes. Every neuron node and every synapse edge is connected to a corresponding parameter node via a so-called *hyper-edge*. Parameter nodes can be shared by neurons or synapses, thus allowing for a possibly massive reduction of data redundancy, which can become extremely important for an efficient handling of such graphs in the working memory of a utilized computer. All parameter nodes are connected via hierarchical edges to the corresponding top-level parent nodes.

The hardware substrate onto which the biological network shall be mapped is also represented in form of a graph. This graph represents the hardware hierarchy, again starting from a full system root node. Depending on the kind of underlying device, the lower hierarchical levels represent single chips (for the `FHW-1` system) or full wafers (for the `FHW-2` system). Sub-units like the network blocks of the `FHW-1` chips or the HICANN modules of the `FHW-2` wafers are represented by own nodes. The neuron circuits are children of these sub-units. At different levels of this hardware hierarchy, the available connectivity infrastructure is incorporated by specific nodes and edges. Nearly all hardware nodes represent hardware-specific configuration space, i.e. parameter memory that has to be filled with reasonable values.

The beneficial idea of the GraphModel mapping approach is that, based on this well defined representation, a mapping algorithm can operate and establish so-called *mapping edges* between neurons in the biological model and neuron circuits in the hardware model. This implies further mapping edges between the corresponding neuron parameter nodes and the parameters intrinsic to most of the hardware nodes. Once all mapping edges have been established by means of an optimization criterion pursued by the mapping algorithm, the transfer of the biological parameter data into the corresponding hardware parameter space can be started. Thanks to the unified graph representation, the optimization scheme of the mapping algorithm can be implemented in a very modular and thus exchangeable fashion, allowing to apply different optimization strategies for different architectures (see Section 3.2.2).

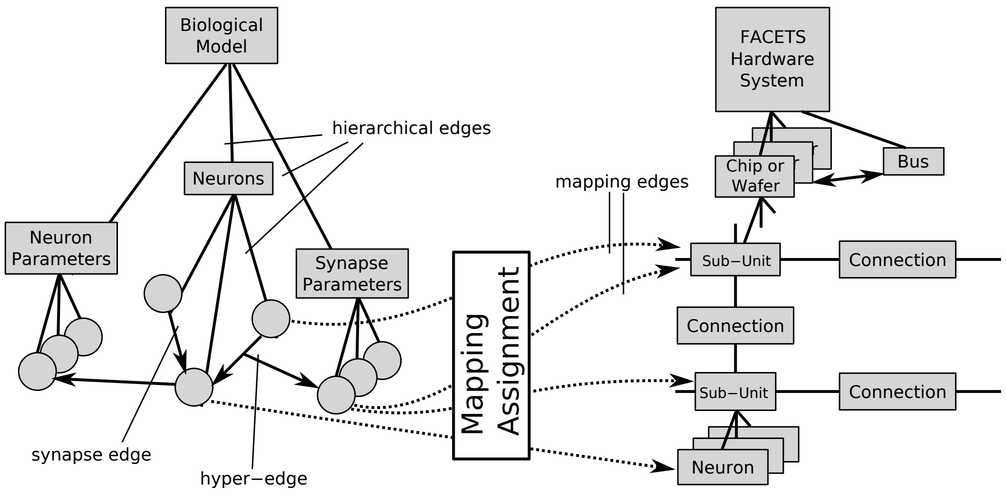


Figure 3.3: Schematic of the topology and parameter mapping strategy between a biological network description and a neuromorphic hardware substrate as developed by K. Wendt (TU Dresden). Both the biological network and the hardware system are represented by hierarchical graphs. In the biological model, neurons and synaptic connections are incorporated as nodes and edges, while the corresponding parameter sets are nodes themselves, connected to their owners via *hyper-edges*. The hardware graph represents the hierarchical structure of the hardware system as close as possible, with each node standing for a hardware unit that provides parameter memory to be filled with reasonable values. Connectivity infrastructure such as routing switches and buses form an essential part of these configurable hardware nodes. A mapping algorithm operates on this graph representation and establishes *mapping edges* between the biological and the hardware nodes by means of an optimization criterion. Figure according to (Müller, 2008).

3.2 Software Architecture

A Yin for the Yang A hardware device that is as complex as the `FHW-1` or the `FHW-2` systems and that is designed to be operated via a connected host computer is useless without an appropriate operating software. Every hardware feature has to be represented by a corresponding software mechanism, otherwise it will not be controllable or observable. Vice versa, such an operating software has to rely on the functionality of certain hardware mechanisms, possibly including worst case scenarios or fluctuation assumptions, otherwise its execution will fail. Hard- and software form one unit, each of both being crucially dependent on the functionality of the other.

In this section, the software framework is presented that has been developed to actually implement the methods and paradigms outlined in Section 3.1. The technologies, third-party software modules and build strategies are summarized in Section 3.2.1. The complete stack of software layers that provide the final integration of the `FHW-1` system into the PyNN framework (see Section 3.1.3) is specified in Section 3.2.2. Additional software modules which extend the core framework are also described, such as higher-level analysis tools (Section 3.2.3), multi-user management (Section 3.2.4), an analog unit-test framework (Section 3.2.5) and a 3D visualization of the mapping from abstract network models to hardware (Section 3.2.6).

3.2.1 Utilized Technologies

The following software technologies are the principal tools utilized for the development and maintenance of the user interface to the `FHW-1` and the `FHW-2` system:

- The programming language *C++* (Stroustrup, 2000).
  This general-purpose, so-called *mid-level* language is an extension of the programming language *C* (Kernighan and Ritchie, 1978). C++ supports procedural, generic and object-oriented programming. It can be used to do both machine-oriented low-level system programming and high-level user application development. It is well suited for complex projects due to, among others, its sophisticated encapsulation methods.

- The programming language *Python* (Rossum, 2000).
  Python is a general purpose high-level language which supports, among others, object-oriented and functional programming paradigms. It has an automatic memory management and a fully dynamic type system. It provides an interactive interpreter and scripting support. Its clear and minimalistic core syntax, its comprehensive standard library, its community-based development model and the resulting large amount of available, well maintained third-party packages make it well suited for the high-level interfacing of neuromorphic systems. See Section 3.1.3 for more application-specific advantages of Python.

- The *Boost.Python* library (Abrahams and Grosse-Kunstleve, 2003).
  Boost is a collection of free, peer-reviewed C++ libraries, many of which are candidates for future extensions of the C++ standard library. Boost.Python provides methods and classes for the interoperability of C++ and Python. This includes a framework for exposing C++ classes and functions to Python (see Section 3.2.2 for a corresponding application).

- The *GNU make* project building tool (GNU).
  The `make` tool is widely used to control and administer the compilation of source code into executables or into libraries which can be used by other programs. It can keep track of changes, inter-dependencies and include hierarchies of source files and control the actual flow of compilation and linking. The `make` tool is not limited to specific programming languages, nor is it limited to compiling code.

- A purchased, proprietary device driver for the communication with a PCI card Jungo Ltd, 2007.

- The *Doxygen* code documentation tool (van Heesch).
  By adding comments with a specific syntax to the source code, Doxygen can extract all these comments and compile a clearly arranged documentation. The format of this output document can be HTML[6] and LaTeX(Lamport, 1994), among others, and in addition to the organized text blocks it generates a hyperlink structure (for the HTML document) and inheritance diagrams. It can be extended by arbitrary extra documentation material not included in the source code. All code presented in Section 3.2.2 is explanatorily commented with this technique, the full extracted Doxygen document is available (see Appendix A.2).

- The *Qt* library (Nokia, 2009).
  Qt is a toolkit for the development of widgets and GUIs. For this purpose, it provides a C++ class library and supports various operating systems and graphics platforms such as X11, Mac OS X and Microsoft Windows. In addition to the graphics development tools, Qt provides features like XML[7] parsing and thread management. Packages exist which allow to utilize Qt via Python (Summerfield, 2008).

- The *OpenGL* specification (OpenGL; GLProgramming) and the *FreeGLUT* toolkit (FreeGLUT).
  OpenGL is a specification for a platform-independent programming interface for 3D visualizations. FreeGLUT implements a library of tools building upon this specification which allow to create and manage windows with OpenGL content.

Except of the proprietary PCI device driver, all of these tools are publicly available and well established in scientific programming.

[6]Hypertext Markup Language
[7]eXtensible Markup Language

3.2.2 Software Layer Stack

In the following, the software layer stacks both for the `FHW-1` system and for the currently developed `FHW-2` system are introduced. The first one is existing as described and has been utilized for nearly all experimental work presented in this thesis. The second one is, like the `FHW-2` system itself, under development.

Operating Software for the Chip-Based System

The implemented software framework for interfacing the `FHW-1` system is structured as illustrated in Figure 3.4. A detailed description of all components depicted in the schematic will be given in the following (see also Brüderle et al., 2009).

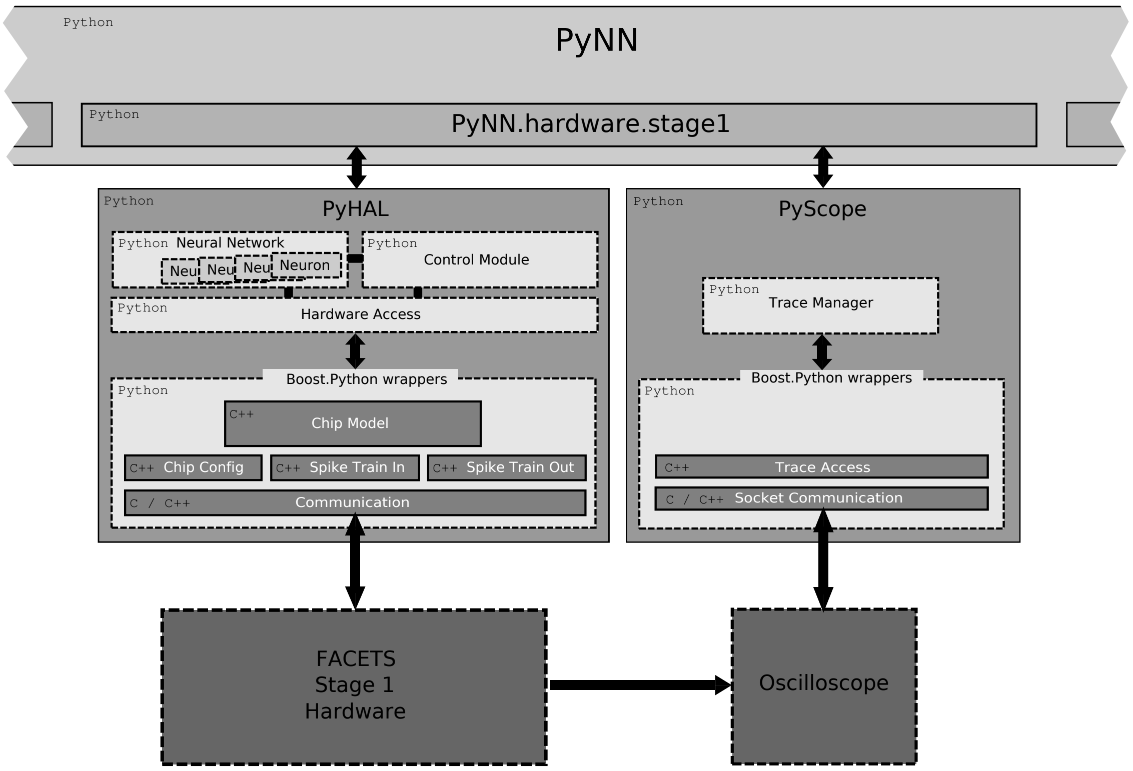


Figure 3.4: Schematic of the software layer stack for the operation of the `FHW-1` system and its integration into the PyNN concept (see also Figure 3.1). See text below for a detailed description of all components.

Digital Communication Stack All digital communication between the host computer and the connected `FHW-1` system, i.e. the configuration of the chip, the sending of input spikes and the retrieving of output spikes, is performed by the following software layer stack:

- At the lowest software layer (in the figure: *Communication*), dedicated C and C++ code (Fieres et al., 2004; Philipp, 2008), which is partly custom-design, partly purchased and proprietary, encapsulates the communication between the host computer and a controller implemented by the Nathan board FPGA (see Section 2.1.5). This includes a stand-alone daemon which provides the transport layer for this communication channel.

- A C++ class, which encapsulates the functionality of the neural network chip respectively its FPGA controller (figure: *Chip Model*), utilizes this communication interface for
  - sending configuration data to the chip,
  - sending stimulation data to the memory located on the Nathan board,
  - initiating the execution of experiment runs, and
  - fetching the output data generated during an experiment from the Nathan memory.
- The complete set of parameter values utilized by this model to configure the chip is encapsulated by an own, dedicated C++ class (figure: *Chip Config*).
- Two instances of the same container class (figure: *Spike Train In* and *Spike Train Out*) comprise the stimulation and the output events handled by the chip model.
- All these low-level C++ classes are exposed to Python via so-called *wrapper classes*. Based on mechanisms provided by the Boost.Python library, these wrapper classes convert value and array types between the two programming languages, but keep their class nature. Only those member variables and functions which are of interest for higher level software layers are interfaced by the wrapping code. If possible, hardware-specific details are hidden at this level by applying reasonable default values and by providing meaningful function and argument names. All wrapped classes are compiled into one single Python module.
- A Python neural network class (figure: *Neural Network*) implements a description of a network of biological neurons. It comprises multiple instances of a Python *Neuron* class, which store all of their parameters and their synaptic connections to or from other neurons. The neural network class provides functions to create, to inter-connect or to delete these neurons.
- A Python control module (figure: *Control Module*) provides user functions which, in a biological terminology, allow to set up the Python neural network representation, i.e. to create and connect neurons, to define the neuron and synapse parameters, and to pass stimulation patterns. The control module can execute an experiment set up like that, and it provides a user function to access the resulting output data generated by the network.
- A Python hardware access module (figure: *Hardware Access*) communicates with these wrapped C++ classes. It fills the chip parameter container with the values it extracts from the Python neural network representation. This filling procedure incorporates the translation from biological to hardware parameters (see Section 2.1.4) and the topology mapping described in Section 3.1.6. The same module also performs the transfer of stimulation data into the C++ containers, including the time transformation described in Section 3.1.5. Furthermore, it transfers all control commands like experiment start signals, and it translates the output data from the wrapped low-level C++ container into its biological representation.

- Together, the following objects form the so-called *PyHAL*[8] module:
  - One Python hardware access module. This module instantiates one object of each low-level C++ class via the Boost.Python wrapper module.
  - One Python neural network object.
  - One Python control module.

  PyHAL implements the Python interface motivated in Section 3.1.3, similar to existing Python interfaces to the software simulators NEST or NEURON.

- A PyNN module for the `FHW-1` system (*PyNN.hardware.stage1*) instantiates and interfaces a PyHAL object. Building upon the functions and biological parameter arguments provided by PyHAL, it implements the full functionality of both the procedural and the object-oriented PyNN API[9] (see Section 3.1.3). The only exceptions are certain synaptic plasticity mechanisms which are not yet mastered for the hardware system (see Sections 4.3.10 and 4.3.11).

Analog Readout Stack In order to access the analog sub-threshold membrane potentials of selected hardware neurons (see Section 2.1.5), a second software layer stack is incorporated into the PyNN module for the `FHW-1` system:

- A C++ routine implements the TCP/IP socket (Braden, 1989) communication (figure: *Socket Communication*) between the host computer and the oscilloscope connected to the hardware system (LeCroy, 2005).

- A C++ class buffers and provides the fetched membrane potential trace data. It also provides functions to write configuration commands to the oscilloscope.

- The C++ code is exposed to Python with the Boost.Python wrapping tool, compiled into one dedicated Python module.

- This module is utilized by a Python membrane trace manager (figure: *Trace Manager*). It automatically performs oscilloscope configurations such that the full trace is acquired with an optimal resolution, i.e. it adjusts time and voltage scales, and it triggers necessary periodical device calibrations. It hides these device-specific procedures and provides a user function which allows to fetch the desired trace data.

- Together, this trace manager and the Python module which wraps the C++ oscilloscope access classes form the so-called *PyScope*.

- The PyScope module is interfaced by the PyNN.hardware.stage1 module for accessing the analog sub-threshold membrane potential data of those neurons which were tagged to be recorded within the PyNN code. The translation from the hardware to the biological voltage domain as described in Section 3.1.5 is performed by the PyNN.hardware.stage1 module, which calls a corresponding translation function provided by the PyHAL hardware access module. This is necessary because the hardware module keeps track of the programmable hardware voltage parameters and the corresponding ranges which are available for the individual chip currently in use (see Section 5.2.1).

[8]Python Hardware Abstraction Layer
[9]Application Programming Interface

Thanks to the implementation of the above software layer hierarchies up to their integration into the PyNN framework, all software modules developed on top of PyNN by the NeuralEnsemble community (see Section 3.1.3), such as analysis and visualization tools, are now available for the `FHW-1` system. An example of a basic PyNN script is given in Section 6.1.2. There, every single line of code is explained in detail, and the script is executed both on the `FHW-1` system and in NEST.

Hardware-Specific PyNN Issues The integration of the hardware interface into PyNN also raises problems. Some of the PyNN API function arguments are specific to software simulators. In the hardware context, they have to be either ignored or be given a hardware-specific interpretation. For example, the PyNN function `setup` has an argument called `timestep`, which for pure software back-ends determines the numerical integration time step. In the PyNN module for the continuously operating hardware, this argument has been re-determined to define the sampling rate of the oscilloscope for membrane potential recordings. Furthermore, the strict constraints regarding neuron number, connectivity and possible parameter values require additional efforts in all software layers mentioned above, i.e. checking for violations and providing instructive warning and error messages. PyNN does not yet support a dedicated framework for fast and statistics-intensive parameter space searches with differential formulations of the changes from step to step, as will be needed to optimize the exploitation of hardware specific advantages (see Section 3.1.1).

Module for Offline Script Testing Without having access to the real hardware system, it is not possible to use the corresponding PyNN module – hence it is not available for public download in its described form. Still, it is planned to provide a modified module on the PyNN website (PyNN, 2008) which allows to test PyNN scripts that are intended to be run on the hardware system. In this context, *testing* means to run the script, but instead of getting back the corresponding network output, the user will get back only all warnings or error messages which would occur with the real system. With such a mapping test module, scripts can be prepared offline for a later, optimized hardware run.

Operating Software for the Wafer-Scale System

Just like the wafer-scale `FHW-2` system itself, the corresponding software stack for its operation is under development. The basic principles and many modules of the `FHW-1` software are adopted with no or with just a few changes. PyNN will be the top-level interface for the definition and control of neural network experiments on this hardware. A software layer stack will provide the transformation of PyNN scripts into hardware configurations and operations. Figure 3.5 shows a schematic of the `FHW-2` software layer stack.

Like for the `FHW-1` system, the core software for the `FHW-2` system is written in C++, while the high-level layers are implemented in Python. As a major difference to the `FHW-1` software stack, the `FHW-2` system utilizes Karsten Wendt's *GraphModel* approach described in Section 3.1.6 for the mapping of network topologies and parameters from the biological to the hardware domain and vice versa.

- The two main components of the GraphModel, a graph representing the biological network (figure: *Bio Graph*) and one representing the full hardware system (figure: *Hardware Graph*) are C++ classes, which both are exposed to Python with dedicated Boost.Python wrapper classes.

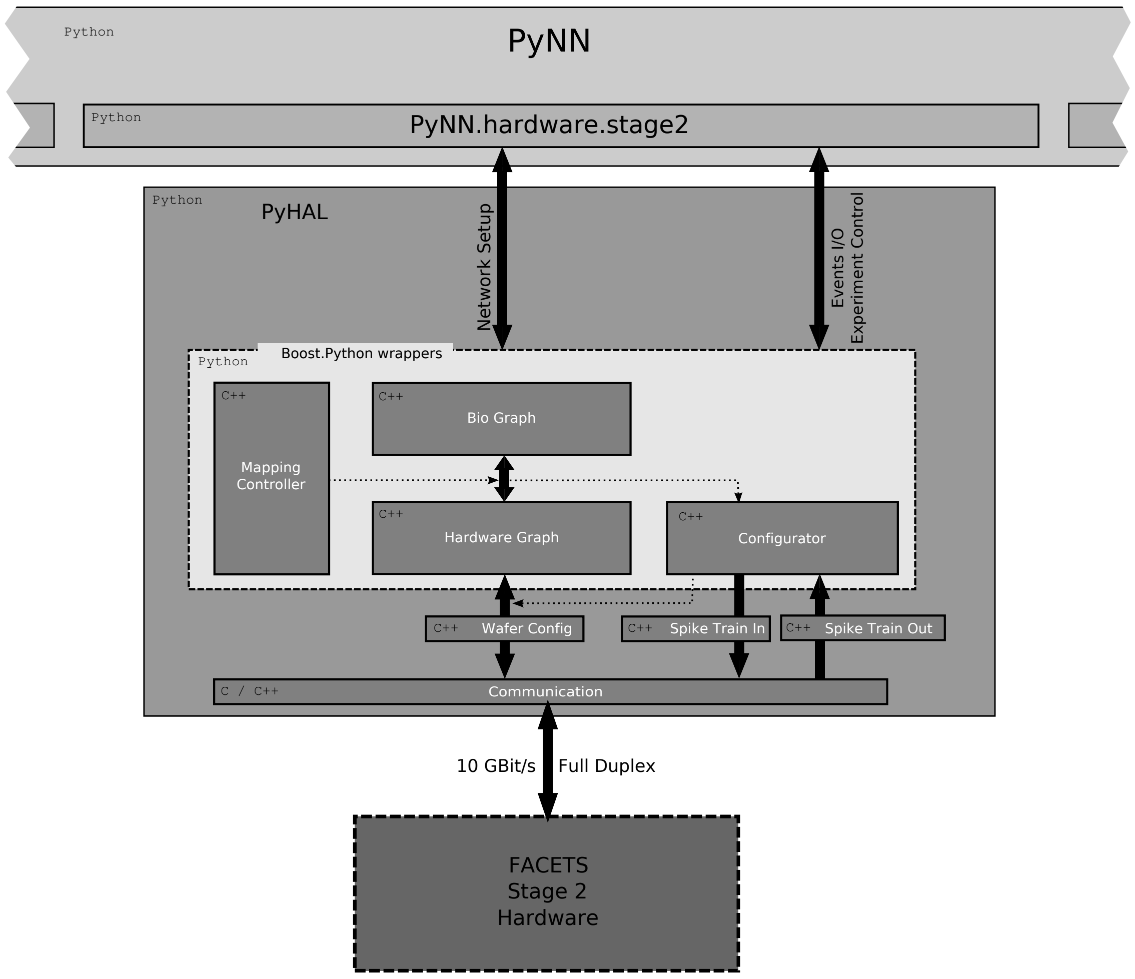


Figure 3.5: Schematic of the software layer stack for the operation of the FHW-2 system and its integration into the PyNN concept (see also Figure 3.1). See text below for a detailed description of all components.

- A C++ class controls the mapping flow (figure: *Mapping Controller*). It determines the sequence of algorithms which are applied in order to establish the mapping edges from biological neurons and parameter sets to hardware neurons and parameter memories (see Section 3.1.6). Thanks to the well-defined structure of the graphs, different optimization techniques can be chosen and exchanged at all levels of the divide-and-conquer mapping approach (Wendt et al., 2008).

- A hardware configuration class (figure: *Configurator*) extracts the configuration data from the hardware graph and initiates the transport of this data to the hardware. The same class performs the translation of the stimulation spike trains from the biological to the hardware domain and the translation of the hardware output spikes into the biological domain, which includes both time transformation and neuron index mapping. For the index mapping purpose, the configuration class needs input from the GraphModel.

- A set of low-level hardware communication modules (figure: *Communication*) provides the transport of all data between the host computer and the hardware system (see Section 2.2), with a bandwidth of 10 GBit/s into both directions.
- Like for the `FHW-1` system, one Python module called *PyHAL* wraps the full complex of C++ classes which perform the biology-hardware translations and the hardware communication.
- Utilizing the functionality provided by PyHAL, a PyNN module for the `FHW-2` system (*PyNN.hardware.stage2*) implements the low-level, procedural and the high-level, object-oriented API of PyNN (see Section 3.1.3).

Neuron Mapping and Synaptic Routing The GraphModel mapping module has to perform essential tasks:

- Create a mapping between biological neurons and hardware neuron circuits.
- Translate the corresponding parameter values and apply calibration data.
- Provide a configuration of the synaptic routing infrastructure on the hardware, such that the resulting connectivity is as close to the biological model as possible.

Due to the limited programmable inter-neuron wiring in hardware on the one hand, but the massive influence of connectivity on the functionality of neural architectures on the other hand, the routing task within the mapping process is of great importance. In a first step, the mapping algorithm places the neurons on the hardware. Typically, this is done such that the number of synaptic connection between the `FHW-2` chip units is minimized, but, in principle, arbitrary optimization criteria can be applied. In a second step, a dedicated routing optimization algorithm developed by Johannes Fieres computes the corresponding configuration of the routing switches.

3.2.3 High-Level Software Tools

As mentioned in Section 3.1.3, interfacing the FACETS hardware systems with Python and PyNN provides many possibilities to accelerate software development cycles, to avoid redundant development effort and to bundle community efforts. A few concrete examples of such beneficial exploitations are described in this section.

NeuroTools

NeuroTools is *"a collection of tools to support all tasks associated with a neural simulation project which are not handled by the simulation engine"* (NeuroTools, 2008).

NeuroTools is written in Python and builds upon the data formats returned by PyNN, although it can be utilized for processing the data generated by any simulation or emulation tool that have a Python front-end. By providing open-source modules for model setup, parameterization, data management, analysis and visualization usable by a broad group of scientists, NeuroTools aims at the increase of analysis tools reliability, at the establishment of best-practices for common tasks and at the avoidance of redundant code development across simulation or emulation communities.

Features of NeuroTools have been used for the experiment setup and data analysis presented in Section 6.2.1.

Graphical Experiment Control

For certain experimental setups, a manual parameter exploration with direct visual feedback of the results can be useful, e.g. for an intuition-guided coarse parameter search in order to reduce the search space for an automated optimization task, or for didactic or demonstration purposes. Especially highly accelerated hardware devices like the `FHW-1` and `FHW-2` systems are predestined to be utilized in such a scenario, since they can return the output to a given experiment setup within periods that are extremely short in terms of human perception. Therefore, a graphical front-end for a given experiment which allows to manipulate a sub-set of parameters, after every change immediately re-runs the experiment and outputs a sub-set of possibly processed observables is sometimes desired.

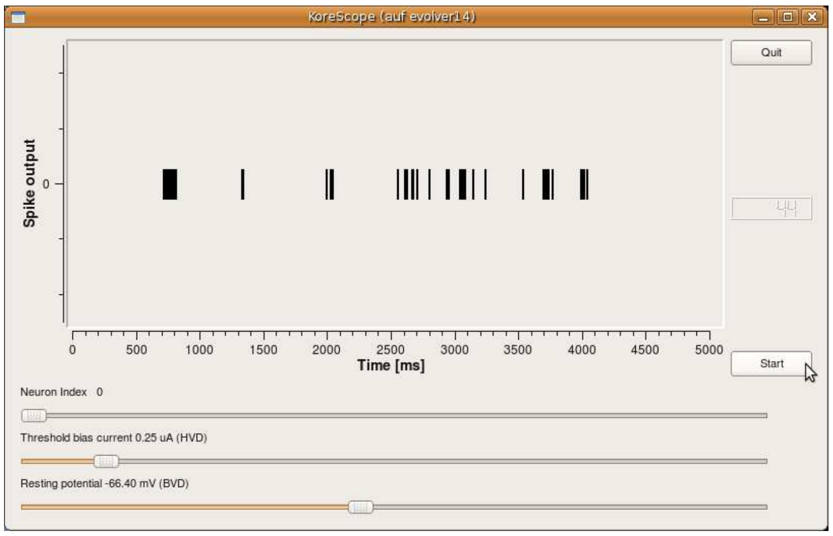
For the Python programming language, third-party packages are available which provide the rapid development and design of graphical dialogs and widgets, with parameter sliders, check boxes, text or number input fields, plot drawing and the like. Such widgets can be started in separate threads and exchange update signals and values with the core experiment. In cooperation with Johannes Bill and Eric Müller, a software framework has been developed which allows to automatically generate a graphical interface to experiments written in PyNN. Arbitrary experiment parameters can be exposed to an interactive manual manipulation via graphical interface tools like the mentioned sliders or check boxes. Figure 3.6 shows an example of such a user widget. It controls a PyNN experiment executed on the `FHW-1` system in a permanently running loop. For a single neuron on the chip, which can be selected with a slider (top-most), the resting potential can be adjusted with a second slider (bottom). Furthermore, the hardware-specific bias current for the on-chip threshold voltage generator is controllable with a third slider (middle). For every run, the resulting output spikes are visualized in an embedded plot.

The purpose of this basic setup is to develop an intuition for the dependency of the firing sensitivity of a neuron on both the written threshold voltage and the corresponding bias current. It has been found that the effective firing threshold of a neuron matches the written value better if the corresponding bias current is chosen high enough. But, as will be described in Section 4.3.2, other objectives limit the available range of bias values. In this context, exploring the qualitative response and firing threshold precision of a neuron depending on these parameters has turned out to be very useful for a deeper understanding of the underlying hardware mechanisms.

Extensive Python Package Usage

Further examples for the beneficial usage of Python third-party libraries in high-level software modules are the following:

- For all kinds of experiments and calibrations, storing data to and retrieving data from files for permanent or temporary storage is done with functions provided by the *Matplotlib* module (Hunter, 2007).
- Routines which handle XML data, needed e.g. for the multi-user management described in Section 3.2.4, utilize available Python bindings for Unix XML handlers.
- All figures presented in this thesis, which contain data acquired with the `FHW-1` system or with software simulators, are generated with the plotting functions provided by the *Matplotlib* module.



Neuron Index 0

Threshold bias current 0.25 uA (HVD)

Resting potential -66.40 mV (BVD)

Figure 3.6: Screenshot of an automatically generated graphical user interface to a hardware experiment set up with PyNN. The experiment is permanently repeated in a fast loop. A single neuron on the utilized `FHW-1` chip can be selected with a slider (top-most). Its resting potential can be adjusted with a second slider (bottom). The hardware-specific bias current for the corresponding on-chip threshold voltage generator is controllable via a third slider (middle). For every run, the resulting output spikes are visualized in an embedded plot. A user of this interface can interactively explore how the firing sensitivity of the chosen neuron depends on arbitrary parameter combinations. The relevance of this setup is explained in the main text.

- Fitting procedures, as e.g. incorporated in the hardware synapse driver calibration routine (see Section 5.2.4), utilize algorithms provided by the *SciPy* package (SciPy).
- The efficient matrix manipulation functions provided by the *NumPy* package are extensively used for all kind of data post-processing throughout the work presented in this thesis, such as the calculation of mean values and standard deviations, matrix addition and multiplication, sorting or slicing. Furthermore, the random number generation features are used in many situations, e.g. to generate the normally distributed threshold values for the experiment presented in Section 6.2.1.

3.2.4 Management of Multiple Users and Systems

As described in Section 2.1.5, with one so-called *backplane*, multiple `FHW-1` chips can be operated at the same time. Currently, multiple of these backplanes exist in the laboratories of the University of Heidelberg, each of them connected to an own host PC, and each of them carrying one or many `FHW-1` chips. One chip on a specific *Nathan* board, plugged into a specific slot of a specific backplane, connected to a specific host PC with a specific PCI card, will be referred to as a *workstation* in this section.

For the parallel operation of all individual workstations, a multi-user and multi-station management framework has been developed. It provides a centrally maintained collection of all workstation information. For every workstation, the following information is stored:

- A unique number which identifies the `FHW-1` chip itself.

- The design version of the chip, e.g. `FHW-1.1`, `FHW-1.2` or `FHW-1.3`.
- The clock frequency with which the chip is operated.
- The paths to all calibration files for this chip.
- The name under which the oscilloscope connected to this chip is identified within the laboratory computer network.
- Identifiers for the output pins and the oscilloscope input channels via which the analog sub-threshold membrane potentials are recorded (see Section 2.1.5) for this individual chip.
- A unique number which identifies the *Nathan* board onto which the `FHW-1` chip is mounted.
- The *backplane slot index* into which the Nathan board is plugged.
- A unique number which identifies the PCI card via which the backplane is connected to its host PC.
- The name under which this host PC is identified within the laboratory computer network (for remote access).
- The name of the default user assigned to this individual chip.

The identifier numbers are used to address the individual devices correctly. All calibration data is chip-specific and therefore needs to be known for every workstation. The operation clock frequency is evaluated for the translation between the hardware and the interpreted biological time domain (`HTD` and `BTD`). Since the oscilloscopes connected to the `FHW-1` systems are separate devices and need to be accessed via dedicated network connections (see Section 3.2.2), they need to be unambiguously identifiable for every workstation, and the mapping between chip output pins and oscilloscope recording channels is essential for a correct data acquisition. The chip-specific user information is evaluated before every experiment run. If the current user of a workstation is not the default one, she will be asked to confirm her operation request. This helps to avoid conflicting accesses to the same workstation by various users at the same time.

3.2.5 Analog Unit Test Framework

In software development, *unit tests* are a common technique to increase and maintain the quality of code. In a software unit test, one specific functional unit, typically a class or a function, is instantiated and executed in a set of fully defined input scenarios. For every scenario, a precise definition of the desired output is provided and compared with the output the program actually generates. Only if the desired output is exactly produced for all input scenarios, the unit passes the test.

Such a rigorous testing paradigm is possible and necessary for digital information processing as performed in von-Neumann-like architectures (von Neumann, 1945), but it is not applicable to the analog processing of continuous values with micro-electronic circuits, as e.g. in neuromorphic hardware systems. As soon as information in an electronic system is represented directly by an analog variable, i.e. without the digitalization concept applied after every processing step, the imperfect nature of the processing substrate makes a perfect control over its behavior impossible. The concept of strict reproducibility of an input-output pair has to be replaced by a scheme which provides tolerances for the output, e.g. by defining levels of maximum deviations.

For the `FHW-1` system, which is subject to different kinds of imperfections (see Sections 4.2 and 4.3), a set of so-called *Analog Unit Tests* (AUT) has been implemented. Those stand-alone micro-experiments extend the idea of common unit tests by features that make them applicable to neuromorphic systems:

- Every AUT utilizes the PyNN module for the `FHW-1` hardware (*PyNN.hardware.stage1*, see Section 3.2.2).
- Within this framework, it provides a minimal, executable experiment setup.
- Every AUT is either a functionality check (*positive* AUT, or *p*AUT) or a hardware malfunction demonstration (*negative* AUT, or *n*AUT). During the development cycles of a chip, which typically requires one or more re-designs and re-productions, especially the negative AUTs are a useful tool to communicate detected malfunctions from the chip users to its designers. They can also minimize functionality verification efforts after a possible design revision. An extensive set of positive AUTs outlines the target design specification in a directly testable and therefore constructive way.
- Every AUT provides information about
  - the applied network architecture, parameters and stimulation data,
  - the recorded variables,
  - the experiment duration,
  - the expected output,
  - optionally: the recommended workstation (see Section 3.2.4), especially for nAUTs,
  - the actually utilized system,
  - the generated output (possibly of multiple trials with the same setup),
  - a precise specification of the software with which it was performed.

  For an easy comprehensibility, essential information about the experiment setup, the recommended platform, the expected and the actually generated output are also provided in a human-readable format.

Section 4.3 refers to multiple AUTs which allow to reproduce various malfunctions of the `FHW-1` system.

3.2.6 3D Visualization of Network Mapping

The *GraphModel* approach presented in Section 3.1.6 performs a mapping between biological neural network descriptions and the corresponding hardware properties. Therewith, it provides not only a neuron index assignment between both domains, but it also generates a translation between the properties of individual neurons and synapses and the corresponding hardware parameter values.

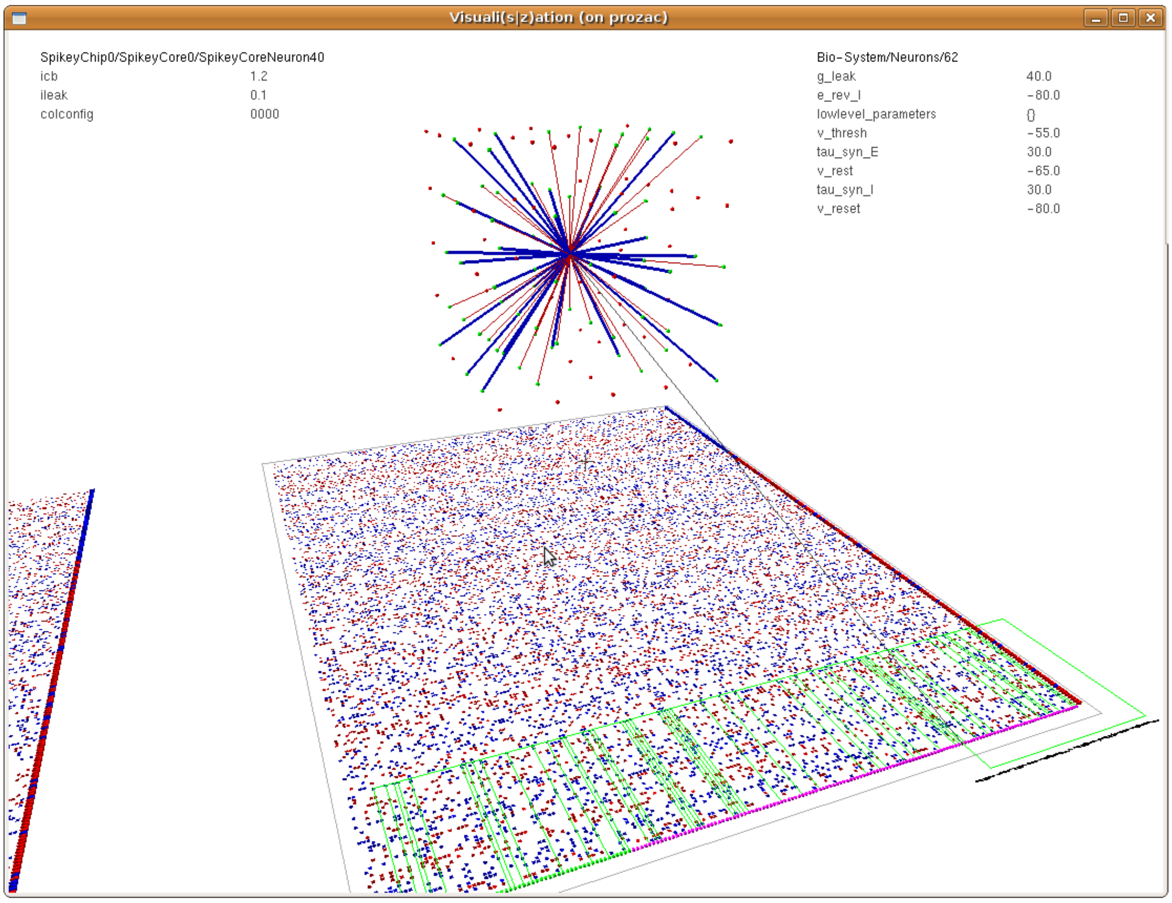
The kind of mapping provided by the GraphModel framework typically is a highly non-trivial multi-objective optimization task, especially for a situation where highly recurrent and structured biological networks are to be mapped to the complex but limited structure of an FHW-1 or an FHW-2 system. The evaluation of a mapping result can be as difficult as its calculation, since possible distortions of the original network have to be detected and assessed.

A visualization software has been developed which provides a way for human users to explore the results of the GraphModel mapping procedure in a virtual 3-dimensional space. The tool can help to detect systematic malfunctions of the applied optimization algorithm, which might be obvious to an intelligent human observer, but hard to grasp with an automatic check routine. Furthermore, the visualization is useful for didactic and presentation purposes of all kind. The current version is only available for mappings to the FHW-1 systems. An extension of the software for a visualization of mappings to the FHW-2 system has not yet been possible due to a lack of finally decided wafer-scale implementation details.

Once the mapping algorithm has generated an assignment between the biological neurons and the hardware cells and has provided a translation between the parameter values of both domains, the visualization can be generated. The Figures 3.7 and 3.8 show two screen-shots of the graphical interface. In Figure 3.7, the biological network is arranged in a cubic grid of $5 \times 5 \times 5$ neurons, located above the 2-dimensional depiction of an FHW-1 network block. The biological cells are drawn as spheres, and their spatial location information is provided by the biological description of the network, e.g. in PyNN (see Section 3.1.3). The hardware neurons are drawn as cubes at the lower edge of the rectangle hardware structure. Their spatial location is automatically generated, such that the full chip structure is drawn close to the biological model, but not overlapping it. The colors of all cells indicate their type, i.e. if they are excitatory or inhibitory. A color coding is also provided for synaptic connections, which are only drawn for those cells which have been selected by a mouse click. The user can select if either incoming or outgoing synapses of a marked neuron shall be visualized. If, e.g. for a typical cortical architecture, all connections were shown at the same time, they would form a very dense structure, annihilating any visual clarity.

In the hardware model (see zoomed perspective in Figure 3.8), the visualized synaptic connections (green lines) connect the neurons with their corresponding synapse drivers (3D triangles, see Section 2.1.2). From there, the connection path is continued via the synaptic nodes, drawn as small rectangles which form most of the large 2-dimensional chip area. Every rectangle codes the weight stored in its synapse node with a color. From these nodes, the connection path is continued to a target neuron.

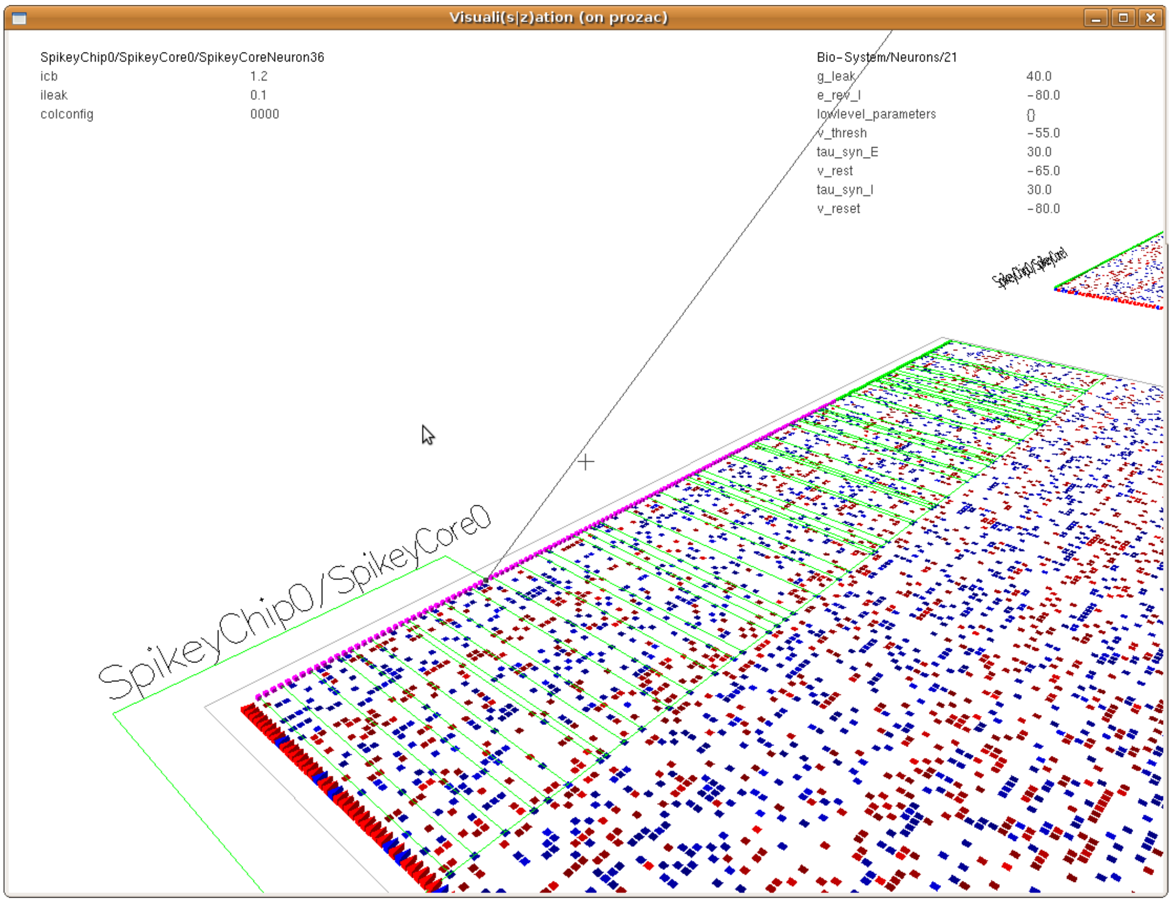
If a cell has been selected either in the biological or in the hardware model, the *mapping edge* between this cell and the corresponding unit in the other domain is drawn (black line), and for both cells all parameters associated with them are printed on the upper left and right corners of the screen. The user can move freely into all directions of the 3D space.



```
SpikeyChip0/SpikeyCore0/SpikeyCoreNeuron40
icb                       1.2
ileak                     0.1
colconfig                 0000
```

```
Bio-System/Neurons/62
g_leak                    40.0
e_rev_I                   -80.0
lowlevel_parameters       {}
v_thresh                  -55.0
tau_syn_E                 30.0
v_rest                    -65.0
tau_syn_I                 30.0
v_reset                   -80.0
```

Figure 3.7: Screen-shot: 3D visualization of the assignment between the $5 \times 5 \times 5$ neurons (spheres) of a biological network and an `FHW-1` network block (large rectangular structure) as determined by the *GraphModel* mapping framework. The hardware neuron circuits and synapse drivers are located at the lower and right edges of the chip structure, while the hardware synapse node array is drawn by small colored patches covering the rectangle area. The hardware and the biological synapses are drawn as lines connecting the sources with the targets. For both the neuron symbols and the synapse lines, the colors indicate if the entity is excitatory or inhibitory. To keep visualization clarity, The synaptic connections are drawn only for single selectable neurons. For marked biological neurons, a mapping edge is drawn to the corresponding hardware unit assigned by the mapping algorithm. Furthermore, the parameters of both the biological and the hardware cell are printed in the upper left and right corners of the screen. For a zoomed view into the hardware graph with more explanations, see Figure 3.8.



```
SpikeyChip0/SpikeyCore0/SpikeyCoreNeuron36
icb                     1.2
ileak                   0.1
colconfig               0000
```

```
Bio-System/Neurons/21
g_leak                  40.0
e_rev_I                 -80.0
lowlevel_parameters     {}
v_thresh                -55.0
tau_syn_E               30.0
v_rest                  -65.0
tau_syn_I               30.0
v_reset                 -80.0
```

Figure 3.8: Screen-shot: A zoomed view into the 3D visualization displayed in Figure 3.7. It shows a corner of the `FHW-1` network block onto which a biological network has been mapped. The cubes in the upper left edge represent the neuron circuits, while the triangle bodies represent the synapse drivers. They span a 2-dimensional area, which is filled by small rectangles, each of which represents a synapse node with a certain weight coded by the drawn color. Green lines illustrate synaptic signal transmission paths that are configured to be enabled. A mapping edge runs in from above, assigning a hardware neuron circuit to a biological cell (which cannot be seen from this perspective).

4 Gaining Control of Chip Functionality and Imperfections

A set of paradigms for the translation between the domain of the accelerated FACETS neuromorphic hardware system (`FHW-1`) and its biological interpretation has been given in Section 3.1. A software framework which provides an executable realization of these paradigms has been introduced in Section 3.2. Building upon this, the following chapter represents a further step towards a neuroscientifically relevant operation of such a neuromorphic device, namely the specification of practical difficulties and problems which occur during its actual operation. Various obstacles such as process-inherent device imperfections (Section 4.2), `FHW-1` prototype flaws and design-related malfunctions (Section 4.3) are described in detail. But in order to evaluate the functionality of certain chip features, possibilities to directly access the interesting variables are missing. In Section 4.1), several techniques are presented, each of which provides an indirect way to acquire the desired information.

4.1 Methods for Indirect Access

Neural behavior can be characterized by a variety of dimensions, e.g. spike rates, membrane potential traces, currents and conductances, amongst others. Depending on the observed system, it can be necessary to deduce magnitudes, which are not or hardly measurable, from easily accessible ones.

E.g. for *in vivo* and *in vitro* recordings, the strength of a synaptic connection typically has to be deduced from the correlation of spiking and membrane activity of its pre- and post-synaptic neurons (Bi and Poo, 1997; Dan and Poo, 2004). The dynamics of multiple ion channels of various types are often combined to one total electric conductance property of the membrane patch they are located on. This conductance can be accessed via patch-clamp techniques, during which voltages are applied and the resulting currents can be measured (Sakmann and Neher, 1995). Another example for hidden variables is the assumed depletion of readily releasable vesicles at synapses in use, a popular explanation for the known phenomenon of short-term synaptic depression (Zucker and Regehr, 2002). In general, models of neuronal and synaptic dynamics often involve variables which are hard to observe *in vivo* or *in vitro*.

If such a model – typically expressed by a set of differential equations – is numerically computed in software simulations, any variable can be accessed arbitrarily. Together with the full flexibility in defining environmental conditions, this model transparency is one main reason for the wide and successful usage of software simulators in modeling neuroscience. Still, the more complex the underlying model becomes, the more computationally expensive its simulation will be. Despite strong efforts towards sophisticated optimization techniques and parallelization of simulations (Morrison et al., 2005), this fact creates a rather slowly receding limit for the range of computable network sizes and experiment durations.

The hardware system described in Section 2.1 provides a way to avoid the scaling problem of pure software simulations. Due to its intrinsic parallelism in neural circuit operation, its speedup factor of up to 10^5 compared to emulated biological time is independent of the size of the implemented network, given that the necessary event communication bandwidths can be provided. Within certain constraints, this is true for the type of devices utilized here (Schemmel et al., 2006; Philipp et al., 2007; Grübl, 2007; Fieres et al., 2008; Schemmel et al., 2008). In fact, a highly configurable event communication is *the* major technological challenge for the design of such highly accelerated large-scale neuromorphic systems.

The software advantages like selectable parameters, definable topologies and adjustable environmental conditions in principle hold for the investigated hardware system either, although in a more limited form. Due to the physical nature of the analog model emulation, most variables and programmable parameters within the system are subject to electronic phenomena like transistor-level variations and noise (see Section 4.2), parasitic leakages, crosstalk or other, design-related imperfections (see Section 4.3).

As a major difference to pure software approaches, the only accessible hardware observables with a biological interpretation[1] are the network's spike output, its membrane potentials and possibly evolving synaptic weights. For example, membranic conductances or currents are not directly accessible, so if necessary, they have to be deduced from the accessible observables. Since for the utilized system the sub-threshold membrane potentials have to be acquired via an oscilloscope connected to the hardware and then need to be integrated into the operating and evaluating software (Brüderle et al., 2007; Davison et al., 2008) via TCP/IP sockets (Braden, 1989), this acquisition channel is rather slow and inefficient. The design of the neuromorphic system has been optimized exclusively for the access to all action potentials generated during an experiment via a fast digital connection. Thus, if possible, a deduction of hidden variables from nothing but the spike output is highly desirable. Purely or partly spike-based methods for specification or calibration will be presented e.g. in Sections 4.1.2, 5.2.3, 5.2.2 and 5.2.5.

4.1.1 Spike-Triggered Averaging on Neuromorphic Hardware

The method of spike-triggered averaging (STA) is well established in neuro-physiology (Thomson and Radpour, 1991; Matsumura et al., 1996; Farina et al., 2001) and modeling neuroscience (Badel et al., 2006; Paninski, 2006). In a typical STA setup, the membrane potential of a neuron plus one or many of its assumed pre-synaptic stimulation spike sources are recorded. Using the spike times of e.g. one of these stimuli as triggers, samples of the recorded membrane potential corresponding to these trigger times are extracted. Averaging over these samples reveals the response of the membrane specifically to the selected stimulus. The schematic shown in Figure 4.1 illustrates this principle.

STA is not limited to membrane potential recordings and its *input* stimuli. Triggering and averaging can be performed with all kinds of events which are possibly correlated with an

[1]Excluding actively controlled parameters, some of which can be read back, too (e.g. the values for the firing threshold V_{thresh}).

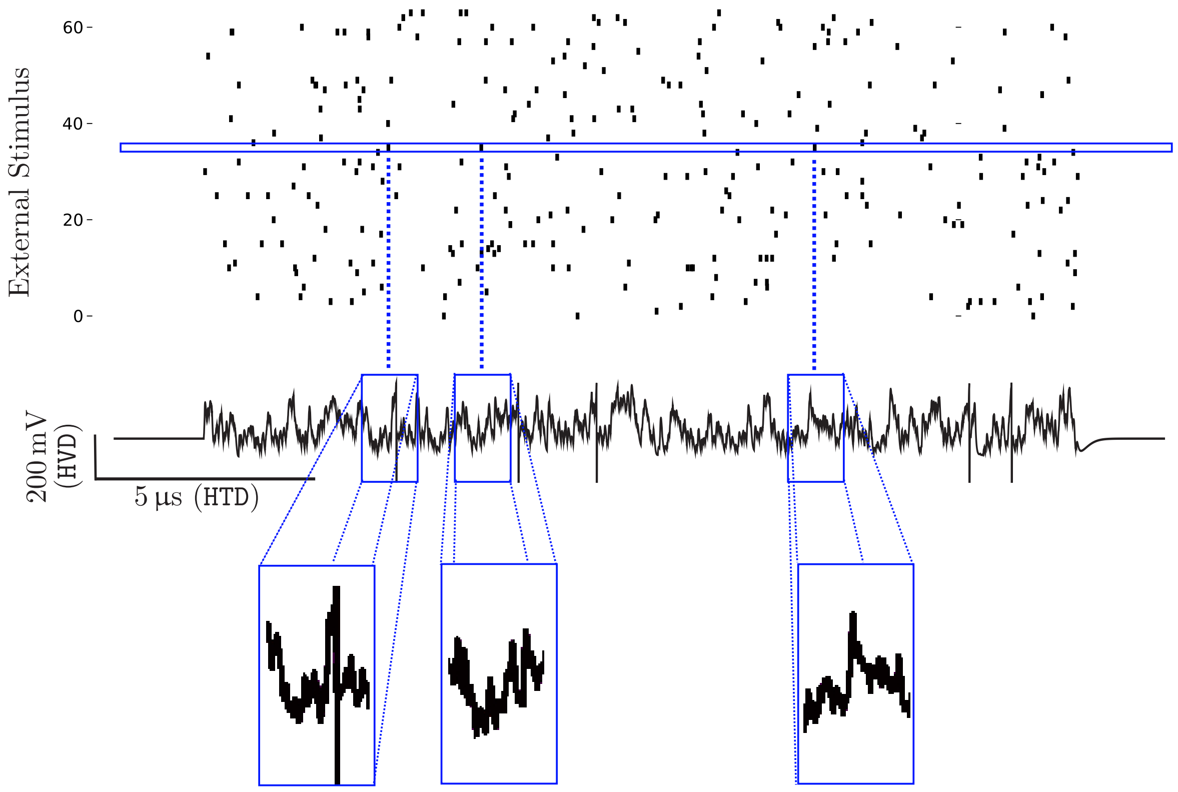


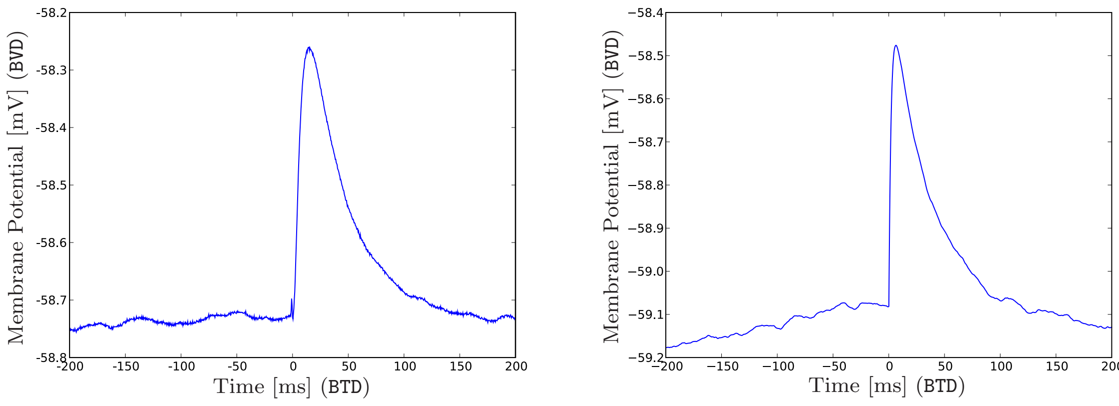
Figure 4.1: Schematic of the spike-triggered averaging (STA) technique used to extract post-synaptic potentials (PSP) from a fluctuating membrane. A neuron is stimulated with externally generated Poisson-type spike trains, its membrane potential is recorded. In order to extract the averaged PSP generated by a specific synapse, the input spikes via this synapse (frame around stimulus spikes) are used as trigger times. Membrane potential samples of a certain length around these trigger times are extracted from the recorded trace (membrane potential zoom boxes). Averaging over many of these samples filters out the membrane potential fluctuations caused by the other synapses and possible noise sources and reveals the pure PSP (see Figure 4.2).

observable of interest.

STA is well suited for being applied to the type of hardware which is in focus of this thesis, because both the input stimuli to a neuron and its membrane potential are available or can be easily retrieved. The same holds for any kind of software simulator supported by PyNN (see Section 3.1.3). Hence, in order to access membrane responses to input spikes, i.e. PSPs, which in a biologically realistic hardware configuration usually are smaller than the noise and activity fluctuation level, STA represents a useful technique and was applied in multiple setups described in this thesis, e.g. for the Sections 5.2.4, 4.3.5 and 4.3.8.

For this purpose a Python class has been implemented which, based on PyNN experiment descriptions, automatically performs STA in hardware or in the software simulator NEST. This can be done for arbitrary membranes and for selectable synapses. The class delivers the averaged trace window in a selectable size and optionally with an exponential fit for the PSP decay. This class has been used for all STA applications within this thesis.

Figure 4.2 shows two typical STA results acquired from a hardware membrane (`FHW-1.3`) and from a NEST simulation. In both cases the neuron is exposed to stimulation from independently firing Poisson spike trains via 208 excitatory and 48 inhibitory weak synapses, each of them firing with 3 Hz (`BTD`), which leads to membrane potential fluctuations in the order of ±5 mV (`BVD`) around a sub-threshold average value (see Appendix A.1 for full parameter set). In both cases, one of the excitatory input channels, i.e. one synapse is investigated with the STA technique. In order to avoid distortions of the PSP image aimed at, the firing mechanism is deactivated during the data acquisition. Sub-figure **(a)** shows the result acquired at a hardware synapse, averaged over 100000 trace sub-samples of 400 ms (`BTD`) length each. Sub-figure **(b)** shows the corresponding NEST results. The excitatory PSPs are cleanly extracted, although their amplitudes of approximately 0.5 mV (`BVD`) are one order of magnitude smaller than the typical synaptically induced fluctuation amplitudes of the membrane signal. The hardware-specific and activity-independent noise, which on the utilized hardware system imposes distortions in the order of 0.5 mV (`BVD`) (see Section 4.2.2), is also filtered down to less than 0.01 mV (`BVD`).



(a) Hardware PSP, average of 100000 samples **(b)** NEST PSP, average of 100000 samples

Figure 4.2: STA technique extracts PSPs caused by single synapses, with amplitudes below noise and activity fluctuation level. **(a)** PSP generated by a synapse driver on `FHW-1.3-No.18`. **(b)** PSP generated by a NEST synapse. The small extra-peak just before the onset of the hardware PSP is digital crosstalk and will be explained in more detail in Section 4.3.8.

4.1.2 High-Conductance State Test

This sub-section incorporates parts of a paper by Bernhard Kaplan, Daniel Brüderle, Johannes Schemmel and Karlheinz Meier, which has been submitted and accepted for publication at the International Joint Conference on Neural Networks (IJCNN) 2009 in Atlanta, USA (Kaplan et al., 2009).

Importance of High-Conductance States for Neuromorphic Systems

One implication of high-conductance states that is especially interesting for the operation of neuromorphic hardware systems is the possible non-monotonous output vs. input rate relation of a neuron. It is described e.g. in (Kumar et al., 2008): If a neuron is stimulated by excitatory and inhibitory input spike trains with the same rate f_{in} each, and if excitation is dominating, then for increasing values of f_{in} the neuron's output rate will also grow. But if the total membrane conductance gets larger due to the stimulus, the impact of the incoming spikes can start to decrease because of a smaller efficient membrane time constant τ_{m} (see Section 1.3.1) respectively shorter and thus less efficient post-synaptic potentials. The phenomenon can be used to generate self-stabilizing states of network activity (Kumar et al., 2008), which has the potential of serving as a mechanism for counter-balancing hardware-specific inhomogeneities and fluctuations (see Section 4.3).

For future biologically realistic experiments on the `FHW-1` and `FHW-2` systems, but also for the basic specification of hardware sub-units, finding a working point in the high-conductance regime is essential. In the following, the effects of synaptic contributions to a neuron's total membrane conductance and consequently, in conjunction with the correlation of the applied input spike trains, on its output spike rate are studied and exploited. A purely spike-based and hence hardware-compatible method is presented which allows to estimate the amount of necessary synaptic stimulation in order to operate within a high-conductance state. Differences and hardware specific advantages of this method compared to a similar one introduced in (Rudolph and Destexhe, 2006) will be discussed in Section 4.1.2.

Spike-Based High-Conductance State Test: Concept

The basic idea of the proposed high-conductance state test is to estimate the total membrane conductance of a neuron by its ability to separate excitatory PSPs which are temporally close. The integration of successive PSPs on a membrane is less likely to cause an action potential if the temporal course of these PSPs is shorter. Assuming fixed time constants for the input-triggered increase and decrease of $g_{\mathrm{e}}^{\mathrm{max}}(t)$, the shape of the resulting PSP is shortened or stretched by the total membrane conductance. Thus, compared to a low-conductance regime, in a high-conductance state successive input spikes have to be temporally closer to cause an increase of the output firing rate, which can be regarded as a better temporal resolution capability of the neuron. In other words, the low-pass filter property of the membrane determines its quality as a coincidence detector.

This makes it possible to deduce the total membrane conductance merely from input and output spike data. Figure 4.3 illustrates the effect of different total membrane conductances on the superposition of PSPs. The same sequence of spikes – a single spike followed by a quadruple – arrives at a relatively slow (solid line) and at a fast (dashed line) membrane. Due to the resulting different temporal courses of the PSPs, those on the slow membrane add up to a larger effective amplitude compared to those on the fast membrane.

Spike-Based High-Conductance State Test: Setup

For testing the input driven responsiveness of a membrane, a single neuron with a constant leakage conductance g_{l} is utilized. In order to vary the externally driven component of the membrane conductance, it receives a set of uncorrelated Poisson spike trains through N_{e} excitatory and N_{i} inhibitory synapses. Each spike train has the same firing rate ν_{in}. The decay time constants τ_{syn} and the maxima $g_{\mathrm{e}}^{\mathrm{max}}$ ($g_{\mathrm{i}}^{\mathrm{max}}$) for the excitatory (inhibitory) conductances are kept constant during all experiments. The aim of the test is to find an average synaptic conductance $\overline{g_{\mathrm{syn}}} \equiv \langle g_{\mathrm{e}}^{\mathrm{tot}}(t) + g_{\mathrm{i}}^{\mathrm{tot}}(t) \rangle$ which, for the utilized membrane, results in a high-

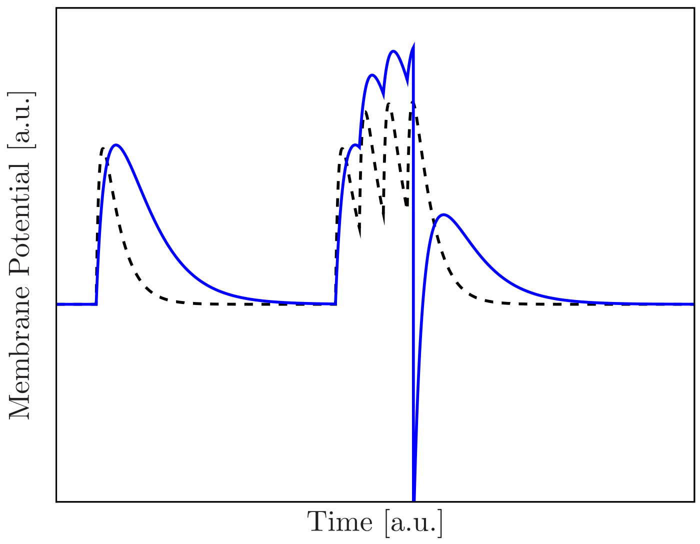


Figure 4.3: NEST Simulation: Overlapping PSPs on a membrane with high (dashed line) and low (solid line) total conductance. Membrane potential and time axis in arbitrary units. For this schematical example, the total membrane conductance has been changed by varying g_l. The shown PSPs result from conductance courses with identical decay times. The conductance amplitudes for both the high and the low conductance membrane have been adjusted such that the amplitudes of the single (leftmost) PSPs become the same in both cases. This results in illustratively different maximum amplitudes of the four overlapping PSPs triggered by identical input spikes. For the fast, i.e. higher conductance membrane, the accumulated potential is not high enough to reach the arbitrarily set spike threshold, while for the slow one it is. The instantaneous membrane depolarization during an action potentials is not modeled in NEST, but the reset mechanism clearly indicates the spike position.

conductance state. For the definitions of $g_e^{tot}(t)$ and $g_i^{tot}(t)$, please see Section 1.3.1. With given values for τ_{syn}, g_e^{max} and g_i^{max}, the temporal integration over N_e excitatory and N_i inhibitory spike trains with firing rate ν_{in} leads to the following average total synaptically induced conductance:

$$\overline{g_{syn}} = \tau_{syn}\,\nu_{in}\,(N_e\,g_e^{max} + N_i\,g_i^{max}) \qquad . \tag{4.1}$$

To control $\overline{g_{syn}}$, the frequency ν_{in} is varied. If all other parameters remained constant, this would result in a corresponding variation of the average membrane potential and, hence, of the output firing rate. An increased output rate deteriorates the responsiveness of the membrane because of the reset mechanism which clamps the membrane potential to the V_{reset} and makes the neuron insensitive to input for a refractory period of $\tau_{ref} \approx 1\,\text{ms}$ (BTD). This could possibly lead to systematic distortions in the results of the proposed method. In order to circumvent such undesired correlations between the average membrane potential, the output rate and the responsiveness of the membrane, the output rate is kept within a limited range ($\nu_{out} = \nu_{target} \pm 0.25\,\nu_{target}$) by changing the ratio N_e/N_i while keeping the sum $N_e\,g_e^{max} + N_i\,g_i^{max}$ constant.

In addition to the Poisson type background, a test stimulus is injected into the neuron via n_{test} excitatory synapses, which consists of periodic packages of n_{pack} equidistant spikes. The period T_{PP} from package to package is kept constant, while the inter-spike interval T_{ISI} within one package is varied from 0 ms to $T_{\mathrm{ISI}}^{\mathrm{max}} \equiv \frac{T_{\mathrm{PP}}}{n_{\mathrm{pack}}}$. With $T_{\mathrm{ISI}} = T_{\mathrm{ISI}}^{\mathrm{max}}$, one package exactly fills T_{PP}. This approach guarantees that the total spike rate fed into the neuron through the test synapses is independent of T_{ISI}. The test stimulus is weakly connected to the neuron, its contribution to the total synaptic conductance (never more than 5%) is neglected in the following. Its absolute contribution to the output firing rate is of no interest, while the *change* in the output rate resulting from the variation of T_{ISI} is evaluated. In this framework, it is not possible to test the response of the membrane without any Poisson background, since the desired output rate cannot be established with the test stimulus only.

Figure 4.4 exemplarily illustrates the test setup: A neuron receives input from Poisson spike trains of a certain frequency ν_{in} (only a subset is shown) and additionally from a test stimulus consisting of packages of equidistant spikes. When the periodic spike packages arrive, the output rate temporarily increases, indicated by an output spike histogram.

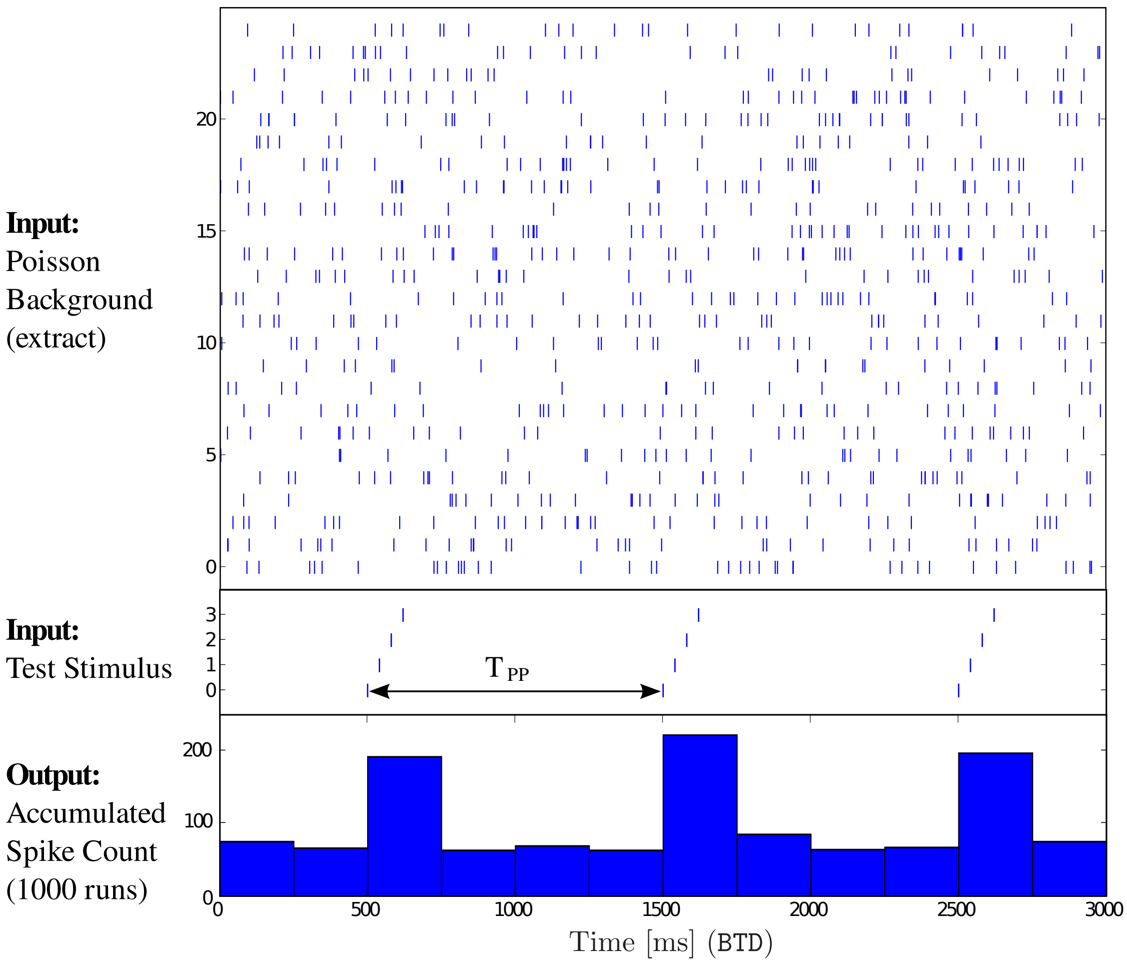


Figure 4.4: NEST Simulation: Example of the spike-based method for membrane temporal resolution evaluation. It is shown the input and the output of a neuron under test. Top: Raster plot of parts of the Poisson background with $\nu_{\mathrm{in}} = 10\,\mathrm{Hz}$ (BTD). Middle: Test stimulus fed into the neuron. Bottom: Resulting output spike count histogram accumulated over 1000 runs.

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

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

To be able to distinguish the output response curves for different conductance states, a characterizing critical quantity τ_{res} is defined as follows: τ_{res} is the critical time interval $T_{\mathrm{PP}}^{\mathrm{crit}}$ between test spikes at which the output rate falls below a certain threshold frequency

$$f_{\mathrm{crit}} \equiv f_{\mathrm{min}} + \frac{1}{2}\left(f_{\mathrm{max}} - f_{\mathrm{min}}\right) \qquad . \tag{4.2}$$

Here, f_{min} is the minimum output rate, i.e. the saturation rate which is reached for a certain value of T_{ISI} and not under-run if T_{ISI} gets larger. The maximum output firing rate resulting from closely coincident input spikes is f_{max}. See Figure 4.5 for an illustration. In (Kaplan, 2008, Section 2.1), a method for the estimation of the error of f_{crit} is given.

Results

The basic set of parameters applied for the software runs is summarized in Table 4.1. The

Neuron Parameters			
Description	Parameter	Unit	Value
Membrane capacitance per 1 mm^2	C_{m}	nF	0.2
Membrane leakage conductance	g_{l}	nS	2.0
Reset potential	V_{reset}	mV	-80.0
Inhibitory reversal potential	E_{i}	mV	-75.0
Leakage reversal potential	E_{l}	mV	-70.0
Firing threshold voltage	V_{thresh}	mV	-57.0
Excitatory reversal potential	E_{e}	mV	0.0
Synapse Parameters			
Synaptic CC decay time constant	τ_{syn}	ms	20.0
Excitatory synaptic CC amplitude	$g_{\mathrm{e}}^{\mathrm{max}}$	nS	0.4
Inhibitory synaptic CC amplitude	$g_{\mathrm{i}}^{\mathrm{max}}$	nS	1.6

Table 4.1: NEST Neuron and synapse parameters (`BVD`, `BTD`).

leakage conductance g_{l} has been chosen particularly low in order to ensure a large membrane time constant for the unstimulated case. If applicable on the hardware system, the other parameters were chosen according to (Muller, 2006), aiming at biologically realistic models. However, in a few cases the values were chosen to better fit the hardware system: For instance, the choice of rather large synaptic time constants reflects hardware limitations, because the chosen speedup factor for the hardware system does support only time constants in the range of 20 to 50 ms (`BTD`, see Section 4.3.5). Furthermore, in order to provide the necessary amount of total synaptic stimulation, the maximum synaptic conductances are large as well, since the number of synapses to a hardware neuron is limited (see Section 2.1). It also has to be noted that neither g_l nor C_m are directly measurable for the hardware. Still, the time constant of the hardware membrane under no stimulation, $\tau_{m,rest} \equiv \frac{C_m}{g_l}$, can be easily measured. By varying a steering current which controls the invisible g_l, $\tau_{m,rest}$ can be calibrated close to the desired value.

Proof of Principle via Software Simulation

In order to avoid that hardware-specific behavior might wrongly confirm the functionality of the proposed method, it is first verified qualitatively utilizing the software simulator NEST. The NEST neuron model and the parameter values are chosen to optimally resemble the hardware. Still, quantitatively equal results from hardware and software are not to be expected, see Section 4.3.

Determining the Temporal Resolution Capability of a Membrane To find the membrane temporal resolution measure τ_{res}, the test stimulus is applied with a package period of $T_{PP} = 1000$ ms (`BTD`). Each package had $n_{pack} = 4$ spikes, with T_{ISI} being varied from 0 ms to 250 ms (`BTD`).

Figure 4.5 shows the result for a background Poisson rate of $\nu_{in} = 4$ Hz (`BTD`), fed into $N_e = 48$ excitatory and $N_i = 51$ inhibitory synapses. The plot shows the expected decrease in the output firing rate with growing T_{ISI} due to the decreasing overlap of PSPs belonging to the test stimulus. Every data point represents the mean value of 250 runs with 10 seconds of simulated time each. The error-bars denote the standard error of the means (SEM). Also shown in the plot is $f_{crit} = f_{min} + \frac{1}{2}(f_{max} - f_{min})$, which is indicated by the dotted horizontal line. The value of T_{ISI} where the curve crosses f_{crit} defines the temporal resolution capability τ_{res} of the neuron.

Background Activity Increases Membrane Temporal Resolution The membrane temporal resolution τ_{res} has been evaluated for various Poisson background rates ν_{in}. Figure 4.6 shows τ_{res} as a function of ν_{in}. The dependence is obvious: The temporal resolution capability of the membrane respectively the coincidence detection property of the neuron improves for higher Poisson background rates. Furthermore, if the synaptic contribution to the total membrane conductance is large enough, the temporal resolution capability saturates. The saturation limit is determined by the time constant τ_{syn} of the synaptic conductance decay.

In the plot, saturation is achieved from $\nu_{in}^{sat} \approx 15$ Hz (`BTD`). In these regions, the membrane potential is nearly immediately following the synaptic stimulation. Since the output rate is dynamically adjusted via N_e and N_i, the critical input rate ν_{in}^{sat} that is sufficient to make τ_{res} saturate also corresponds to two critical values N_e^{sat} and N_i^{sat}.

Hence, following Equation 4.1, the temporal resolution saturation can be quantitatively

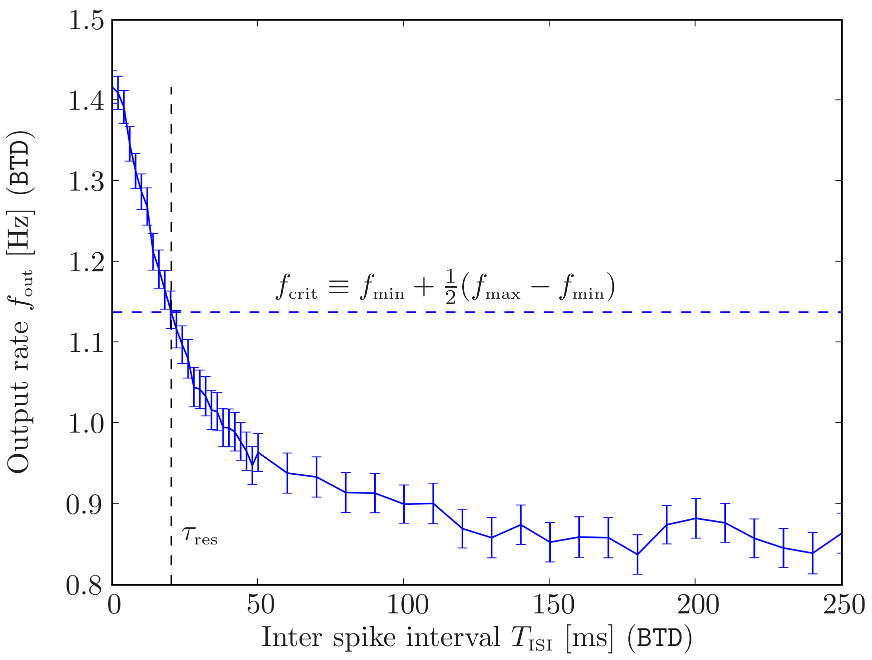


Figure 4.5: Neuron output firing rate vs. inter-spike interval of applied test spikes. The horizontal dotted line shows f_{crit}. f_{min} is defined by the mean output rate for T_{ISI} in [150 ms, 250 ms] (BTD). The vertical dotted line indicates the temporal resolution τ_{res} defined in the text.

estimated to be reached with the minimum amount of synaptically induced conductance

$$\overline{g_{\mathrm{syn}}^{\mathrm{sat}}} = \tau_{\mathrm{syn}}\, \nu_{\mathrm{in}}^{\mathrm{sat}} \left(N_{\mathrm{e}}^{\mathrm{sat}}\, g_{\mathrm{e}}^{\mathrm{max}} + N_{\mathrm{i}}^{\mathrm{sat}}\, g_{\mathrm{i}}^{\mathrm{max}} \right) \qquad . \tag{4.3}$$

For this example with $N_{\mathrm{e}}^{\mathrm{sat}} = 44$, $N_{\mathrm{i}}^{\mathrm{sat}} = 52$ and $\nu_{\mathrm{in}}^{\mathrm{sat}} = 15\,\mathrm{Hz}$ (BTD), an average synaptic conductance of $\overline{g_{\mathrm{syn}}^{\mathrm{sat}}} \approx 30\,\mathrm{nS}$ (BVD) is needed to reach a maximum responsiveness of the membrane. This is a more than ten-fold increase compared to the pure leakage conductance g_{l}.

In (Brette and Gerstner, 2005), the transition of a membrane into the high-conductance state is defined by a ratio of 5:1 between its total conductance and its pure leakage conductance. The input rate which is necessary to create this amount of total conductance for the described experiment is indicated by a dashed line in Figure 4.6.

High-Conductance States in Silicon

As indicated above, variations from the pure software results are to be expected, since the hardware is subject to electronic phenomena like noise, crosstalk, parasitic capacitances and leakages. Many of the issues described in Section 4.3 introduce dynamics distortions that are hard to be quantified and hence cannot be perfectly mapped to the available standard neuron models in NEST or to custom extension. Furthermore, the conductance courses generated in the hardware synapse drivers are implemented as increasing and decreasing *currents* which control physical conductances to the corresponding reversal potentials, see Section 2.1.2. The wires and switches for these currents have capacitances which impose a loss of synaptic efficacy for low input rates plus an activity dependent low-pass filtering effect.

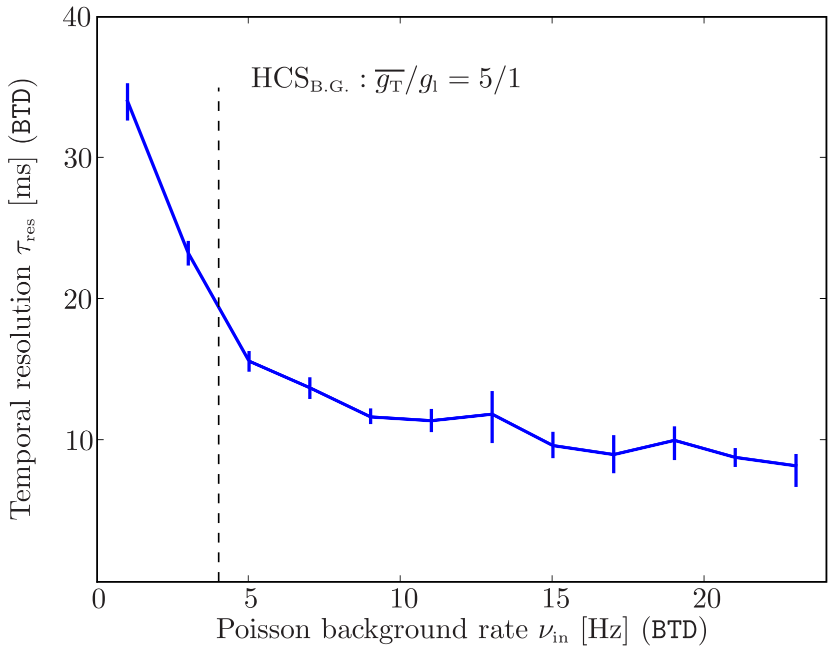


Figure 4.6: NEST Simulation: Temporal resolution τ_{res} of a membrane plotted versus the applied background input rate ν_{in}, acquired with the method proposed in Section 4.1.2. Note the saturation for input rates larger than approx. 15 Hz (`BTD`). The vertical dashed line represents the input frequency necessary to generate a high-conductance state according to the definition given in (Brette and Gerstner, 2005).

Thus, a theoretical prediction for the transition to a high-conductance state as defined in (Brette and Gerstner, 2005) and as indicated in Figure 4.6 cannot be made for this system. Despite these obstacles, the proposed test method provides the possibility to experimentally find reproducible high-conductance regimes on the hardware system.

All hardware experiments for the presented high-conductance state test are performed on the `FHW-1.3-No.17` system. For the hardware model, the neuron parameters are chosen identically to the software simulations (see Section 4.1.2) if possible. As stated above, for a hardware neuron circuit the absolute value of g_{l} cannot be directly accessed, but the membrane time constant in rest can be measured. A hardware control current which determines g_{l} is set to a value very close to the minimum of the available range, resulting in a membrane time constant of $\tau_{\mathrm{m,rest}} = (16 \pm 3)\,\mathrm{ms}$ (`BTD`).

The hardware synapse parameters were set such that the desired output rate of the neuron could be kept throughout the covered input rates. The chosen weights roughly correspond to biological quantal conductance increases of $g_{\mathrm{e}}^{\mathrm{max}} \approx 1.5\,\mathrm{nS}$ and $g_{\mathrm{i}}^{\mathrm{max}} \approx 6\,\mathrm{nS}$ (`BVD`), with decay time constants of $\tau_{\mathrm{syn}} = (27 \pm 8)\,\mathrm{ms}$ (`BTD`). These values have to be interpreted with care, since, due to a hardware malfunction, especially the excitatory synaptic efficacies are activity dependent (see Section 4.3.4 for details).

Figure 4.7 shows the result of the proposed high-conductance state test conducted on a hardware neuron with the given parameters. Like in Figure 4.6, the decrease of τ_{res} as a function of ν_{in} and the saturation from a certain input rate can both be observed.

Analogously to Section 4.1.2, the critical amount of synaptic conductance can be determined for this hardware neuron. In the shown example setup, saturation is reached from $\nu_{\text{in}}^{\text{sat}} \approx 17\,\text{Hz}$ (BTD), with $N_{\text{e}}^{\text{sat}} = 28$ and $N_{\text{i}}^{\text{sat}} = 27$. Together with the values for $g_{\text{e}}^{\text{max}}$, $g_{\text{i}}^{\text{max}}$ and τ_{syn} and provided an ideal hardware system, the necessary average synaptically induced leakage to put the hardware neuron into a maximally input sensitive regime is, in its biological interpretation, $\overline{g_{\text{syn}}^{\text{sat}}} \approx 94\,\text{nS}$ (BVD). But once more, due to the variety of hardware specific issues mentioned above and in Section 4.3, which distort both $g_{\text{e,i}}^{\text{max}}$ and the experiment result $\nu_{\text{in}}^{\text{sat}}$ itself, this biological interpretation is not valuable. Still, it is a very useful information for experimentalists who want to prepare a high-conductance state in hardware.

The difference compared to the NEST model is assumed to be caused mainly by the loss of excitatory synapse driver efficacy in cases of high load on the excitatory reversal potential, and by the capacitances of the wires which route the synaptic conductance courses, which also distort the synaptic impact on the neuron in efficacy and time, depending on the input rate. This capacitance probably also explains the large values of τ_{res} for small background rates, where the given values for τ_{syn} and $\tau_{m,\text{rest}}$ suggest smaller results – especially since in the NEST reference experiment τ_{res} decreased very fast with growing synaptic stimulation. But as can be seen from the data, it is still possible to find a level of background stimulation which is sufficient to put the neuron into a high-conductance state.

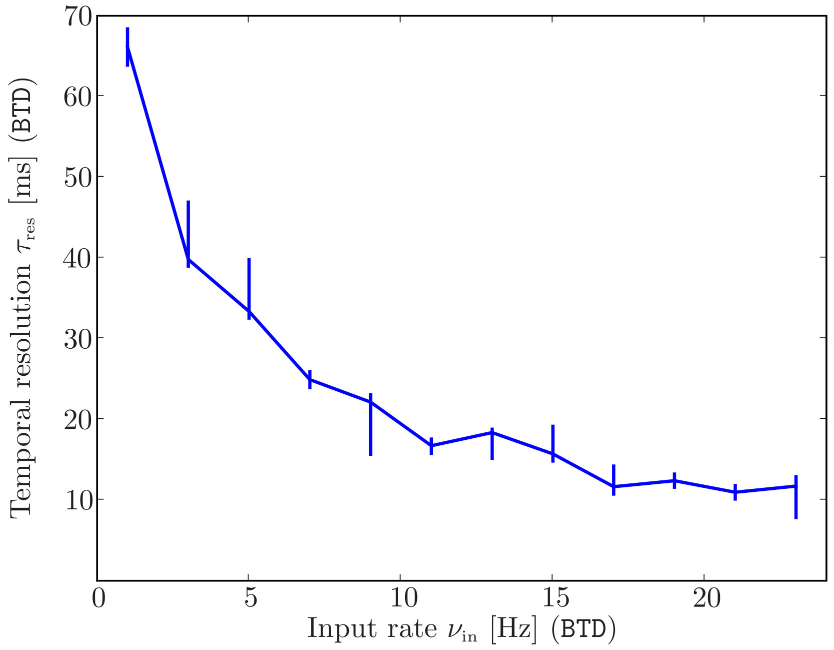


Figure 4.7: Hardware: Temporal resolution τ_{res} of a VLSI membrane plotted versus the applied background input rate ν_{in}. Saturation is reached at approximately 17 Hz (BTD).

In Section 6.1.3, the presented method is used to actually specify the necessary amount of synaptic stimulation for a larger set of neurons on a FHW-1.3 system, providing a representative estimator for future experiments which require high-conductance states.

High-Conductance State Test: Discussion

For a neuromorphic hardware system, a spike-based method has been presented which allows to find the amount of synaptic stimulation necessary for a neuron to operate in a high-conductance state. In order to avoid possible misleading hardware-specific behavior, our approach has been first tested by pure software simulations. Subsequently, the method has been applied to the hardware system, and the results clearly demonstrate the functionality of the proposed technique. Compared to possible alternatives based on e.g. sub-threshold analyses via oscilloscope, this method is faster and can be robustly automated. Both speed and robustness are particularly important for the hardware system, since the presented high-conductance state test has to be applied for several neurons in order to find a reliable setting that is valid for the whole chip. This is necessary because leakage conductances towards the resting potential, parasitic leakages towards reversal potentials and parasitic capacitances affecting post synaptic currents' shapes can differ from circuit to circuit.

A similar method applied to neuron models in software is described in (Rudolph and Destexhe, 2006) and is also spike-based, but has a few draw-backs compared to the presented one. This is mainly due to portability issues for the hardware platform, which does not offer the flexibility for artificial setups as software simulators do. First, the method in (Rudolph and Destexhe, 2006) requires in the order of 30 synaptic inputs which synchronously generate very strong conductance courses. This is hard to be adopted for the presented system, because the hardware platform is limited in terms of both number of synaptic inputs and the available range and reliability of synaptic efficacies. Additionally, sending more than 4 perfectly synchronous spikes into one neuron at a time is not possible on the utilized chip, although the technically necessary and automatically generated fan-out of input spike times in the order of tenths of biological milliseconds is probably negligible. Sweeping arbitrary input firing rates is not possible either, the proposed ranges of up to 150 Hz (`BTD`) exceed the hardware bandwidths. Furthermore, the method proposed in (Rudolph and Destexhe, 2006) brings difficulties for automation due to the fact that it requires the detection of peaks in inter-spike interval histograms which often are ambiguous. An algorithm needs to find the height of a peak that is not necessarily the global, but just a local maximum within the histogram, and can be determined only by its expected location. Finally, the fact that the measure τ_{res} suggested in this paper is given in millisecond dimension represents an intuitive quantity for a neuron's temporal resolution property.

One drawback of the presented approach is that changing the ratio of excitatory to inhibitory stimuli in order to keep the chosen output target rate constant makes it difficult to apply the proposed method to many neurons at the same time – which would have made the presented test faster. Furthermore, the functionality of the method is dependent on the output firing rate: If the output rate without test stimulus $\nu_{\mathrm{out}}^{\mathrm{nostim}}$ is chosen too large, the firing rate with applied test stimulus and hence the difference rate $f(T_{\mathrm{ISI}}) = \nu_{\mathrm{out}}^{\mathrm{stim}}(T_{\mathrm{ISI}}) - \nu_{\mathrm{out}}^{\mathrm{nostim}}$ would not decrease monotonously - as seen in Figure 4.5 - and thus finding the critical quantity τ_{res} would not be possible. This can be interpreted as a firing sensitivity which is too high, resulting in a critically high probability that the first test spike within a package already triggers an output spike and pulls the membrane to the reset potential. This would effectively reduce the impact of the following spikes due to the reset mechanism and thus has to be avoided.

Nevertheless, the spike-based high-conductance state test provides an automatable tool for tuning neuromorphic systems towards an input-sensitive regime, even although the direct measurement of conductances is impossible. The principle might help to avoid extra wiring and analog-to-digital converters in future neuromorphic designs, because the indirect conductance access method during a system specification phase can be sufficient. Thus, prototypes could exploit more of the available chip area.

4.1.3 Long-Term Plasticity

To test the STDP functionality of the chip, the measurement technique described in the following was applied. Results of investigations on larger arrays of hardware synapses, which make use of this method, are presented in Section 6.1.5.

Within each hardware synapse of the `FHW-1` system, correlation flags are provided which can be periodically checked by the digital STDP controller (see Section 2.1.3). If enough charge has been accumulated on one of the capacitors designed for the local summing of correlation information, the flag z_O will be set in order to indicate that a change of the digital weight is to be performed. The flags z_c and z_a provide information about the dominating correlation type: causal or acausal. The developed method enforces correlations between pre- and post-synaptic activity and then reads back the resulting values of z_O, z_c and z_a, while weight changes or flag resets by the STDP controller are disabled.

To artificially generate correlations in a controllable way, a pre-synaptic spike has to be sent into the *observed synapse* while the corresponding post-synaptic neuron has to be forced to fire with a certain, deterministic temporal distance before or after that arrival time. As explained in Section 2.1.3, each temporal correlation between pre- and post-synaptic spikes charges a specific capacitor, and if the charge on one of these capacitors is *large* enough, the corresponding flag will be set by the detection circuit. Normally, multiple correlated spike pairs are necessary to cause a correlation occurrence flag to be raised.

The basic idea of the measurement method is to decouple pre-synaptic input spikes from post-synaptic firing, i.e. to avoid a necessarily causal correlation. Instead, multiple strong synapses – which the observed one is not part of – trigger the post-synaptic firing (*trigger synapses*). The *observed synapse*, which receives the pre-synaptic spike, is set to its minimum weight value in order to minimize its effect on the post-synaptic neuron. The post-synaptic spike, which is triggered by the strong synapses, is released with a controllable time offset of Δt with respect to the pre-synaptic spike. Thus, if Δt is negative, a seemingly causal correlation between pre- and post-synaptic firing is created, otherwise an acausal one. Figure 4.8 shows a schematic of this hardware-specific STDP measuring method.

Still, to trigger a single output spike at a fixed time, various constraints of the hardware have to be considered: First, as a single synapse driver typically is not strong enough to initiate a post-synaptic firing, multiple strong synapses have to be used for the triggering of the post-synaptic spike. Second, only one input spike per synapse driver FIFO buffer and time bin can be delivered, thus only one out of 64 synapse drivers may be used for triggering. As 256 synapse drivers are served by a total of four event input FIFO queues, three trigger synapses with maximum weight plus one observed synapse with minimum weight are utilized.

A spike train S_base is generated which consists of equidistant events. This stimulus is transmitted to all trigger synapses synchronously, while the *observed synapse* receives the same spike train with a temporal shift δt. Thus, the post-synaptic neuron repetitively receives three strong trigger EPSPs[2] plus one weak EPSP via the synapse of interest. Consequently,

[2]Excitatory Post-Synaptic Potential

the neuron fires a sequence of spikes caused only by the trigger EPSPs. The correlation flags, which are possibly raised in the observed synapse due to the resulting correlations of input and output spikes, will not be reset until the full input spike train has been applied, then they are read out.

As stated above, a single correlated spike pair is normally not sufficient to raise a correlation flag. Thus, the number N_{pairs} of equidistant spikes within S_{base} is iteratively increased from run to run until it is large enough to reliably raise the flag. The acquired number N_{pairs} reflects the shape of the underlying STDP modification function F: It is inversely proportional to the absolute of $F(\Delta t)$ (where Δt is the sum of δt and a configurable, hardware specific offset), and the sign of $F(\Delta t)$ is determined by the type of correlation, i.e. if it is causal or acausal. If N_{pairs} exceeds a certain, configurable limit, $F(\Delta t)$ is defined to be zero. The current algorithm implementation receives a minimum value Δt_{min}, a step size t_{step} and a maximum value Δt_{max}, and it scans $F(\Delta t)$ based upon these parameters.

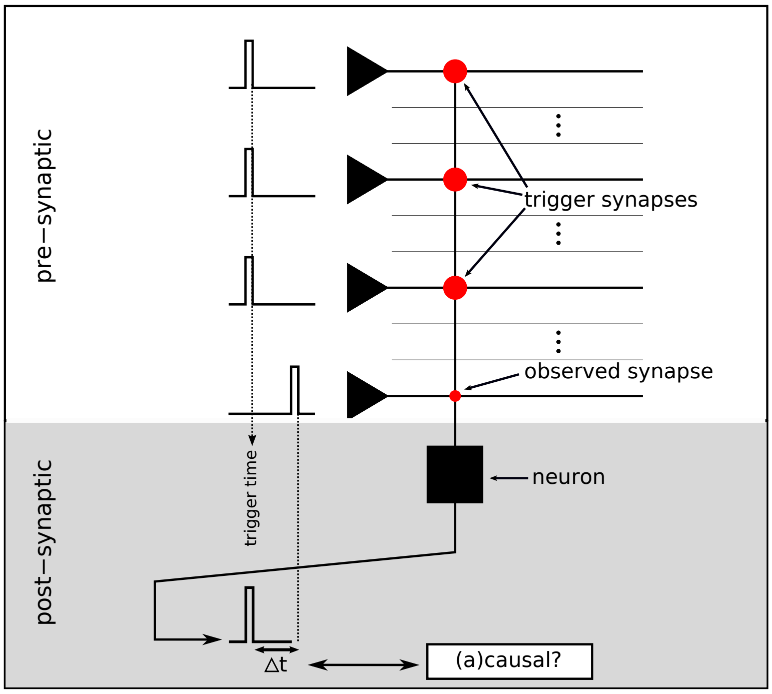


Figure 4.8: Principle of indirect STDP modification function access: Three spikes via strong excitatory so-called *trigger synapses* force a neuron to fire. The *observed synapse* the modification function of which shall be extracted receives a fourth spike, shifted by an adjustable time Δt against the trigger time. Hence, at the observed synapse, a pre-/post-synaptic spike correlation will be registered which can be adjusted via Δt. After multiple applications of such artificially generated correlations, the observed synapse will raise the corresponding flags which then can be read by a controlling software. Figure according to (Müller, 2008).

In order to find valid STDP configuration regimes for the hardware, multiple combinations of the parameters which control the amplitude of the causal branch of the modification function $F_{\mathrm{c}}(t)$, the amplitude of its acausal branch $F_{\mathrm{a}}(t)$ and the correlation threshold $V_{\mathrm{thresh}}^{\mathrm{STDP}}$ have to be swept (see Section 2.1.4).

Furthermore, as the hardware neurons do not respond homogeneously when excited by the three trigger input spikes due to reasons explained in Section 4.3.4, a weight calibration routine has to be applied. The weights of the trigger synapses are optimized to cause exactly one post-synaptic spike per trigger event. More algorithmic improvements accelerate the measurement: First, to detect if it is possible to find correlations for a given Δt at all, the routine starts with a high number N_{pairs} of correlated spike pairs, and if no correlation flag is detected, $F(\Delta t)$ is set to zero. Second, multiple neurons are stimulated at the same time, allowing to test multiple synapses in parallel. Finally, probably due to one or more of reasons listed in Section 4.3.11, multiple scan repetitions and averaging are necessary in order to achieve significant and clean results.

The data presented in Figure 4.9 has been acquired on a `FHW-1.1` synapse with the principles described above. The plot shows the inverse of the number N_{corr} of correlated spike pairs that are necessary to raise a z_{O} flag, plotted versus the time difference Δt between the applied pre- and post-synaptic spike.

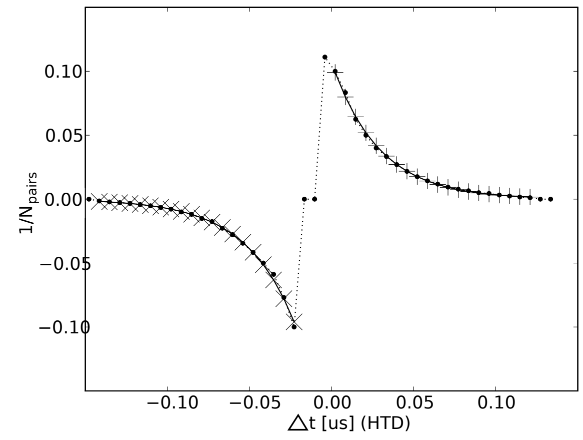


Figure 4.9: An STDP modification function of a `FHW-1.1` synapse, extracted with the method described in this section. It plots the inverse of the number $\mathrm{N}_{\text{pairs}}$ of pre-/post-synaptic spike pairings that are necessary to raise a correlation flag, versus the time difference Δt (`HTD`) between the correlated spikes.

In Section 6.1.5, the described method is applied to an array of synapses. The resulting modification curve arrays illustrate problems with the systematic control and specification of STDP curves, which are explained in Section 4.3.11.

4.1.4 Membrane Time Constants

In order to measure the membrane time constant $\tau_{\text{m}} = \frac{C_{\text{m}}}{g_{\text{l}}}$ of a hardware neuron, the membrane potential $V(t)$ has to be set to a value that differs from its resting potential V_{rest} such that the subsequent exponential decay back to V_{rest} can be observed. One way to achieve membrane potential offsets is via synaptic stimulation, i.e. by applying temporary conductances towards the excitatory or inhibitory reversal potential. But since synaptic conductances have a significant impact on the effective membrane time constant (see Section 1.3.1), and since the conductance courses generated in the hardware synapses are temporally stretched (see Sections 2.1.2 and 4.3.5), synaptic stimulation is not a good choice. After applying a synaptic stimulus, one would have to wait until the synaptic conductance course has definitively ended, but during this period the achieved membrane offset will already have decayed significantly.

Instead, a τ_m access method has been developed together with Andreas Grübl, which follows a simple, but efficient principle. No recording of the membrane potential itself is necessary, the method requires only the spike output of the analyzed neuron:

The firing threshold V_thresh of the analyzed neuron is set below its resting potential V_rest:

$$V_\mathrm{thresh} = V_\mathrm{rest} - \frac{1}{e} \cdot (V_\mathrm{rest} - V_\mathrm{reset}) \qquad . \tag{4.4}$$

Consequently, the neuron fires permanently with a frequency determined only by τ_m and its refractory period τ_ref. Each time the firing threshold has been exceeded, the membrane potential $V(t)$ is pulled to its reset potential V_reset, where it remains for the time τ_ref. As soon as the refractory period has passed, $V(t)$ starts to evolve back towards V_rest by means of an exponential decay. Due to the specific choice of V_thresh, the time between the end of the refractory period and the next threshold crossing is exactly τ_m. Hence, the resulting firing rate of the neuron is $f = 1/(\tau_\mathrm{ref} + \tau_\mathrm{m})$, i.e. the membrane time constant can be extracted from the acquired firing rate as follows:

$$\tau_\mathrm{m} = \frac{1}{f} - \tau_\mathrm{ref} \qquad . \tag{4.5}$$

Figure 4.10 illustrates the setup for this indirect access principle.

Since the membrane potential recording for `FHW-1` neurons has to be provided via an oscilloscope (see Section 2.1.5), which significantly slows down the data acquisition process, the presented, purely spike-based access method is well suited for this system. A membrane time constant calibration routine based upon this principle is presented in Section 5.2.3.

Note that this measurement method requires a high precision in the configurability of V_rest, V_thresh and V_reset, and that the value of τ_ref needs to be known well. Due to design-inherent malfunctions presented in Section 4.3.2, these conditions are not fulfilled for the first three versions of the `FHW-1` system. Nevertheless, the method will be applicable in future systems which solve the responsible problems, and for the `FHW-1` devices currently at hand, a modified routine already works (see Section 5.2.3).

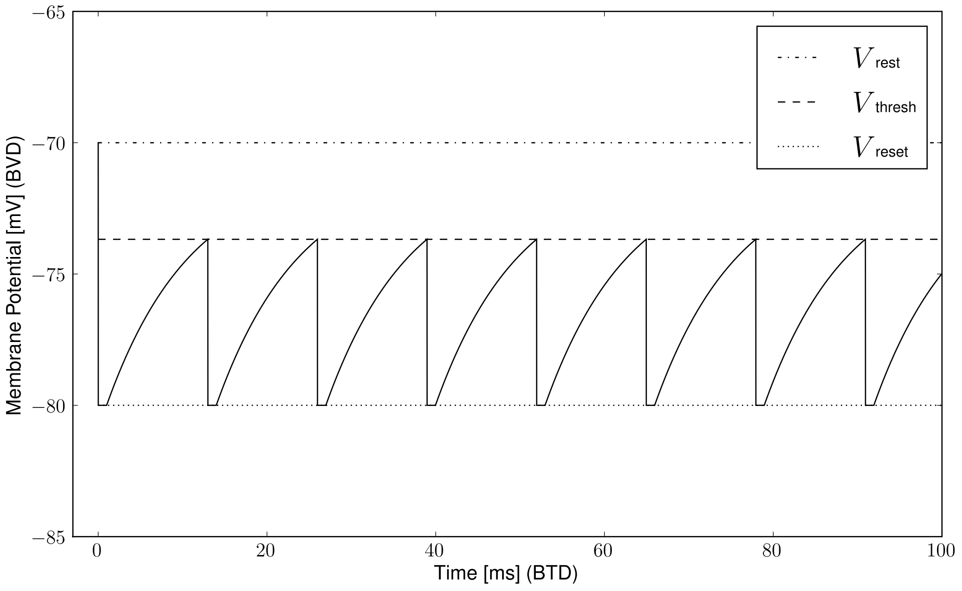


Figure 4.10: Illustration of the method to access hardware membrane time constants τ_{m}. In order to avoid distortions of τ_{m} by synaptic conductances, a specific setup is chosen: The firing threshold of the analyzed neuron is set to $V_{\text{thresh}} = V_{\text{rest}} - 1/e \cdot (V_{\text{rest}} - V_{\text{reset}})$, i.e. below its resting potential V_{rest}. Consequently, the neuron fires permanently with a frequency determined only by τ_{m} and its refractory period τ_{ref}. After every crossing of the firing threshold, the membrane potential $V(t)$ is pulled to the reset potential V_{reset}, where it remains for the period τ_{ref}. As soon as the refractory period has passed, $V(t)$ starts to evolve back towards V_{rest} by means of an exponential decay. Due to the specific choice of V_{thresh}, the time between the end of the refractory period and the next threshold crossing is exactly τ_{m}. The resulting firing rate of the neuron is $f = 1/(\tau_{\text{ref}} + \tau_{\text{m}})$.

4.2 Process-Inherent Imperfections

In the following, sources of hardware imperfections are described which, for an engineer who utilizes a commercially available hardware technology, are unavoidable due to different reasons. In Section 4.2.1, variations in the physical structure of the hardware substrate, caused by production imperfections, are described. Those variations are static in the sense that they stay unchanged once a device is produced. Dynamic aspects of circuit behavior imperfections, i.e. the temporally evolving deviation of electronic variables from their ideal values, or just *noise*, are introduced and discussed in Section 4.2.2. Note that the static variations explained in Section 4.2.1 a sometimes referred to as *fixed-pattern noise*.

4.2.1 Hardware Production

During the production of CMOS VLSI devices, not all process conditions can be kept perfectly constant (Dally and Poulton, 1998, Section 6.5.3). Typical parameters which are used to quantify process variations are the threshold voltages of the produced *n*- and *p*-type MOSFETs[3], i.e. the critical gate voltages beyond which the corresponding transistors are in a state of a maximum conductance between the source and the drain terminals. Given a certain voltage V_{DS} between these terminals, the resulting current I_{DS} that flows between them is a further examples of such process parameters. For these variables, Table 4.2 gives estimators of process variations that are to be expected from the 0.18 µm process utilized for the production of the `FHW-1` and the `FHW-2` systems[4].

Parameter	Typical	On-Chip	Between Chips
n-type transistor threshold	0.42 V	±7 mV	±50 mV
p-type transistor threshold	−0.44 V	±9 mV	±50 mV
n-type transistor drain-source current I_{DS}	6.3 mA	±100 mA	±900 mA
p-type transistor drain-source current I_{DS}	−2.4 mA	±30 mA	±350 mA

Table 4.2: Variation of CMOS production process parameters for the 0.18 µm process which is used for the `FHW-1` and `FHW-2` systems. All values are estimators for transistors at a temperature of 20 °C, with a gate width of 10 µm, a gate length of 0.18 µm and a drain-source voltage of $V_{\text{DS}} = 1.8\,\text{V}$. The "On-Chip" column provides values for typical variations of the parameters from transistor to transistor on the same chip, while the "Between Chips" column gives the corresponding variations for transistors located on different chips from different wafers.

Usually, the threshold voltage deviations of *n*-type and *p*-type transistors from their typical values are correlated, which results in a limited 2-dimensional space of possible deviation combinations. Chip production foundries specify this space by so-called *corners*, which are worst-case estimators for all possible combinations of extraordinarily fast and slow *n*-type and *p*-type transistors.

The effect of such process variations on the functionality of the circuits that are made of these transistors has to be estimated individually for every specific circuit. Especially in the

[3]Metal-Oxide Semiconductor Field-Effect Transistor

[4]In such a process, 0.18 µm is the horizontal extension of the smallest realizable structure. Due to a non-disclosure agreement, only coarse estimators for the process variations are provided here.

case of transistors that play a crucial role in the control of very small currents, deviations from the typical specification can have a large relative impact on the controlled variable. For small designs, worst-case analyses can be computed by simulating the full circuit on basis of the provided process corners. But due to limited computational power, for large and complex designs such an analysis can still be performed only component-wise.

In case of the `FHW-1` devices (see Section 2.1), process-inherent transistor variations significantly influence the behavior of the modeled neurons and synapses, as will be shown in Sections 4.3 and 5.2.

4.2.2 Electronic Noise

Every electronic signal that can be acquired from VLSI devices is subject to different kinds of noise. The concept of digitalization is one possible approach to reliably avoid a resulting loss or distortion of the processed information. But not all tasks to be solved by microelectronics are realized best with a digital design, which usually requires more physical space, consumes more power and is possibly slower than a corresponding dedicated analog circuit. The accelerated FACETS neuromorphic hardware systems are so-called *mixed-signal* devices, i.e. they are composed of both digital and analog circuitry. They exploit the benefits of both design paradigms for the different stages of information processing. The digital domain of such a chip mainly performs the exchange of data and control signals between the digitally operating external interface devices (see Section 2.1.5) and the analog domain of the chip. The purely analog part of the chip physically implements the neuron and synapse models described in Section 2.1.2. The analog approach allows to do this in a very efficient way in terms of consumed chip area, dissipated power and speed (Schemmel et al., 2006, 2007).

Types of Noise

A variety of physical effects causes different kinds of noise (Dally and Poulton, 1998, Chapter 6). The most important types relevant in the context of this thesis are:

- So-called $1/f$-noise, which is a typical phenomenon observed in many physical processes, e.g. if the intensity of a noise signal depends on the kinetic energy of a particle or mechanism. In CMOS transistors, this type of noise is significant and renders a quality criterion of the production process.

- Thermal (or Johnson-Nyquist) noise, which is caused by the thermal agitation of charge carriers. It occurs even if no currents flow, and can be reduced by cooling the affected circuitry. It is so-called *white noise*, i.e. the noise power is uniformly distributed over the frequency spectrum. For thermal noise, the distribution of its amplitudes, acquired over arbitrary time steps, is Gaussian.

- Shot noise, which is caused by the discrete nature of charge. If the number of charge carriers that pass a potential barrier per time is small, the fluctuations of this statistical process result in observable fluctuations of the current. Like for thermal noise, the power density of shot noise over its frequency is constant, i.e. it is white noise, and its amplitudes are normally distributed.

- Noise caused by activity in surrounding circuitry during the operation of a device, often referred to as *deterministic noise*. Although this kind of signal distortions is design-related, it has noise character, because in practice, it is nearly impossible to fully predict its effects and to fully avoid it by design. A typical example is so-called *crosstalk*, i.e. signal distortions resulting from capacitive or inductive coupling between neighboring circuits. Another effect of this category is the activity-dependent change of supply voltages.

Noise on a Hardware Membrane

A basic example of noise affecting the `FHW-1` circuits is shown in Figure 4.11: The histogram of 25000 membrane potential samples recorded from an unstimulated `FHW-1` neuron over 100 µs (`HTD`) is plotted. A Gaussian fit with a width of $\sigma(V_{\mathrm{rest}}) = 1.4\,\mathrm{mV}$ (`HVD`) is added to the figure and matches the distribution well. A priori it is not obvious if the source of this

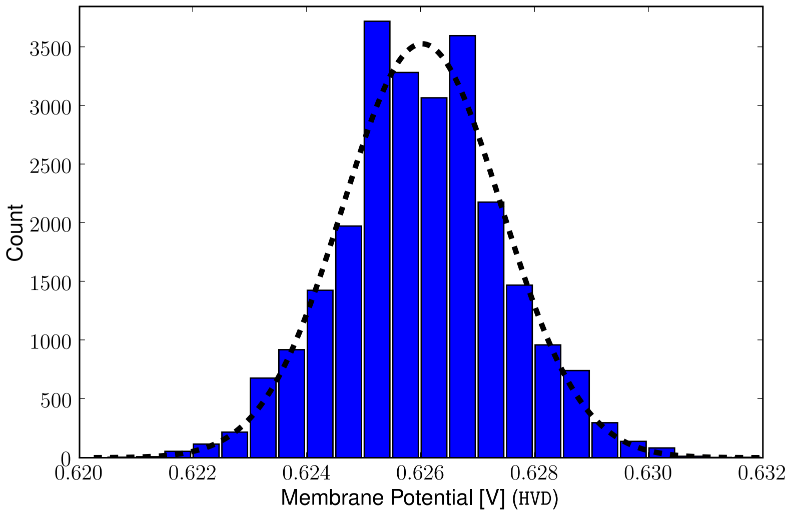


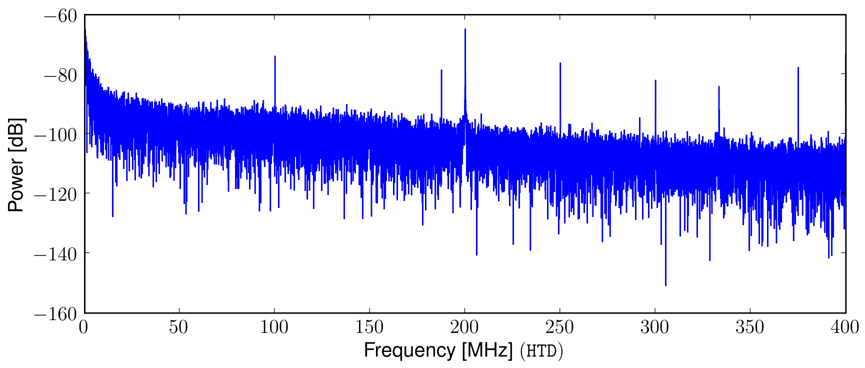
Figure 4.11: The membrane potential of a single unstimulated `FHW-1` neuron is recorded over 100 µs (`HTD`) with a sampling period of 4 ns (`HTD`). The distribution over the resulting 25000 samples is plotted (histogram), and a Gaussian fit is added to the data (dashed line): $\mu(V_{\mathrm{rest}}) = 625.8\,\mathrm{mV}$ and $\sigma(V_{\mathrm{rest}}) = 1.4\,\mathrm{mV}$ (`HVD`). In the biological interpretation of this voltage distribution, the width corresponds to $\sigma(V_{\mathrm{rest}}) \approx 0.1\,\mathrm{mV}$ (`BVD`).

noise is generated on the chip or if it is imposed by the utilized readout devices, i.e. by the oscilloscope and the connecting cable. Therefore, the same measurement is performed with the power supply for the `FHW-1` system switched off. An acquisition of the same number of voltage samples via the same pin with the same device results in a voltage value distribution width of $\sigma(V_{\mathrm{off}}) \approx 0.1\,\mathrm{mV}$ (`HVD`). Hence, it is assumed that the main contribution of noise on the membrane recordings from an activated neuron is generated on the chip. It is still not clear if this noise is really a membrane property or if it is imposed during the routing of the membrane voltage to the readout pin. But since the corresponding width of the voltage distribution in the biological voltage domain is only approximately 0.1 mV (`BVD`), and since the hardware is subject to other, design-related malfunctions that introduce imprecisions on much larger scales (see Section 4.3), a low priority can be assigned to this issue.

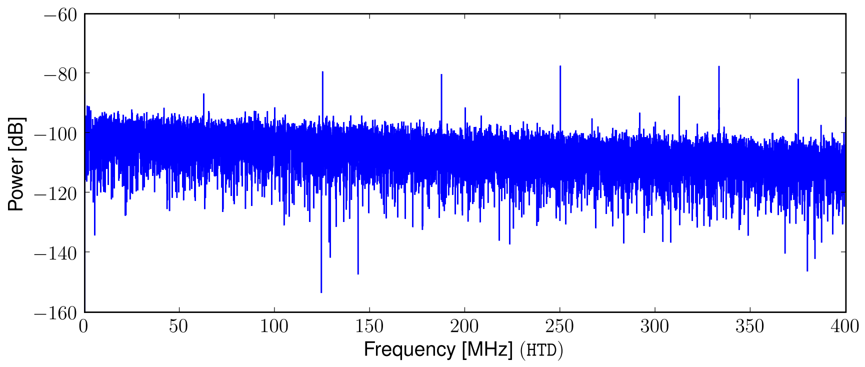
Power Spectrum Figure 4.12 shows two power spectra: Sub-figure **(a)** plots the power distribution against the frequency of the membrane potential trace of an unstimulated `FHW-1` neuron placed on a device with power supply. Sub-figure **(b)** is acquired with exactly the same setup, but with the power supply of the full `FHW-1` system switched off, i.e. mainly the noise of the readout chain is acquired.

In the spectrum of the activated neuron, peaks at the frequencies 100 MHz, 200 MHz and 300 MHz can be observed, which is assumed to be introduced by crosstalk of digital activity onto the recorded analog variable. This assumption is motivated by the fact that the full system is operated with a frequency of $f_{\text{chip}} = 100\,\text{MHz}$. This frequency is chip-internally doubled to $f_{\text{int}} = 200\,\text{MHz}$ (see Section 4.3.9), and based upon this internal clock, the event transformation between the analog and the digital part of the chip is performed. These crosstalk peaks are not present in the spectrum of the readout noise. In both spectra, though, peaks at $f \approx 190\,\text{MHz}$, 250 MHz and 330 MHz are visible. It has been found that the noise at these frequencies is introduced by the oscilloscope device itself, i.e. the source of these peaks is off the chip.

For frequencies larger than 20 MHz (`HTD`), there is a clear $1/f$-dependency in the noise spectrum of the active membrane, while this effect is weaker in the pure readout device spectrum. In case of the active membrane, strong contributions to the noise occur for frequencies smaller than 20 MHz (`HTD`). A closer investigation of the lower band of the power spectrum, which is not shown here, reveals that the major fraction of noise power is contributed by frequencies even below 1 MHz. This kind of noise is assumed to be introduced mainly by the power supply, which is not perfectly stable.



(a) Electrical power supply on.



(b) Electrical power supply off.

Figure 4.12: Sub-figure **(a)** shows a power spectrum of a membrane potential trace acquired from an unstimulated `FHW-1` neuron. The system is operated with a chip clock of $f_{\mathrm{chip}} = 100\,\mathrm{MHz}$. Corresponding peaks, probably caused by crosstalk from the digital part of the chip to the recorded analog signal, can be observed at 100 MHz, 200 MHz and 300 MHz. Sub-figure **(b)** shows a power spectrum acquired with the same setup, but with the electrical power supply for the `FHW-1` system switched off, i.e. the observed noise is dominated by the readout device chain. As expected, the 100 MHz peaks are absent. For further interpretations, see main text.

4.3 Prototype-Specific Malfunctions and Design-Related Interferences

In the following, multiple issues will be described that have been found to impose constraints and limitations for working with the FHW-1 system. This list is the result of a systematic effort to gain control and understanding of the hardware's behavior and characteristics. In most cases, technical explanations have been found and will be provided. If the reason for a problem is not yet understood, it will be described phenomenologically.

For multiple issues, calibration mechanisms (see Section 5.2) or workarounds (see Section 5.1) have been developed that minimize or eliminate the negative impact on the usability of the system. Although all issues that are known at the moment of writing are mentioned in this section, and although it is unlikely that a major problem has been overlooked, the list does not claim completeness. Please read Section 2.1 for a thorough understanding of the parameters and dynamics that will be referred to in the following. Unless otherwise expressly mentioned, the described problems have been observed on all instances and all versions (FHW-1.1, FHW-1.2 and FHW-1.3) of the chip.

Naming Conventions

A *neuron block* is the set of 64 adjacent neuron circuits on an FHW-1 chip that share the same output FIFO Buffer. Accordingly, a *synapse driver block* is the set of 64 adjacent synapse drivers that share the same input FIFO Buffer.

4.3.1 Spike Recording Deadlocks

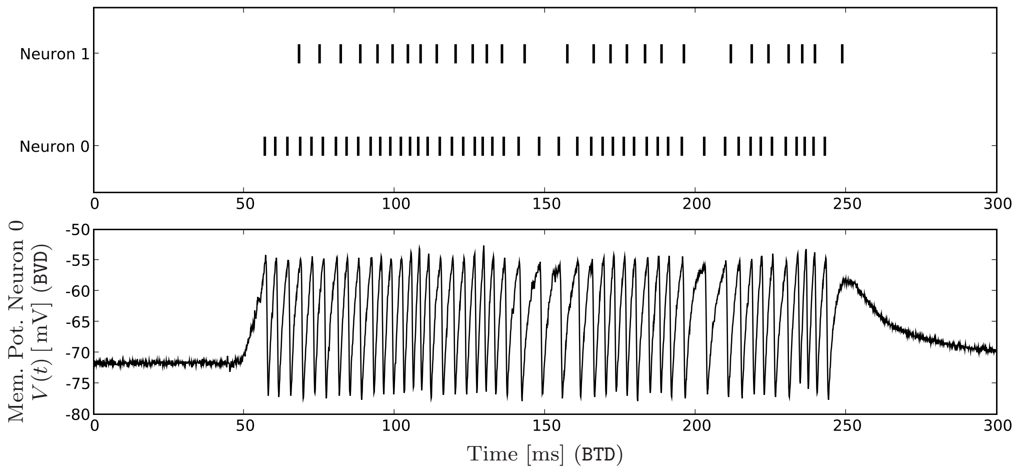
When recording from more than the last n_{rec} neurons of a neuron block, output spikes from different neurons that are generated temporally close can result in a total spike readout deadlock, i.e. no spike is recorded from this block anymore. There is no signal provided by the hardware system that would indicate such a deadlock, so it is not obvious whether a neuron mutes because it is just not stimulated sufficiently or a deadlock has occurred. So far, no spontaneous end of this deadlock situation within one experiment has been observed, only a so-called *neuron reset* (see Section 5.1.1) can resolve the deadlock.

For all FHW-1 systems, the critical number n_{rec} has been observed to vary from 5 to 9, depending on the specific chip and neuron block. For most blocks, applying $n_{\mathrm{rec}} = 8$ avoids the deadlocks. In the following, the set of neurons that can be recorded at the same time without the danger of deadlocks will be called REC. The precise conditions for such deadlocks, e.g. in terms of synchrony or firing rates, are not yet known, but the problem can already occur if only two neurons outside of REC fire with rates of around 1 Hz (BTD). The higher the output rate of two or more neurons outside of REC is, the sooner the deadlock will happen. If only one neuron is recorded per neuron block, no deadlocks occur, no matter whether or not this neuron is an element of REC.

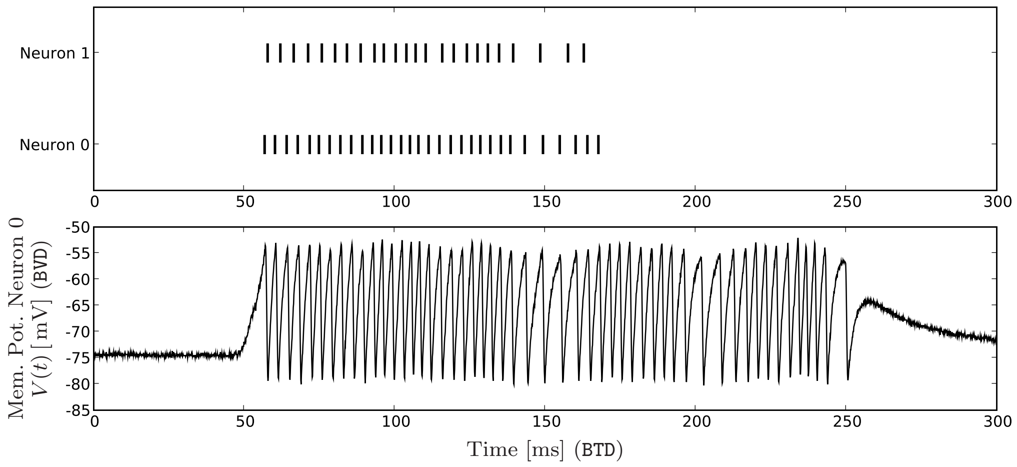
Figures 4.13**(a)** and **(b)** show a normal and a deadlock situation on the FHW-1.3-No.18 system: Two neurons are exposed to the same strong excitatory stimulation from $t = 50\,\mathrm{ms}$ to $t = 250\,\mathrm{ms}$ (BTD), but are not connected to each other. They start to fire shortly after the onset of the stimulus. The spikes of both neurons and the membrane potential of one of them are recorded. As can be observed, the voltage-recorded neuron is continuously crossing the firing threshold and then pulled back to its reset potential, just as expected. Figure 4.13**(a)** shows a situation in which both neurons are located within `REC`, hence the digital recording of the fired spikes works well throughout the whole run. For the recordings shown in Figure 4.13**(b)**, both neurons are chosen from outside of `REC`, so a deadlock situation occurs: After around 170 ms (`BTD`), from both neurons no spikes are received anymore, although the analog behavior seems to be unaffected.

By applying so-called *neuron resets*, the deadlocks can be resolved temporarily. See Section 5.1.1 for more information on that.

The precise reason for the deadlocking bug is design-inherent and not yet fully understood. Further simulations of the responsible circuits including a detailed model of parasitic effects are required.



(a) Both neurons are located within REC: No recording deadlock.



(b) Both neurons are located outside of REC: Recording deadlock occurs.

Figure 4.13: In the upper sub-figures of **(a)** and **(b)**, the recorded spikes of two neurons during a phase of strong stimulation are shown. The membrane potential of neuron 0 is plotted in the corresponding lower sub-figures. In **(a)**, no deadlock situation arises because the two neurons are located within REC, i.e. they can be recorded at the same time. In **(b)**, where both recorded neurons are located outside of REC, a readout deadlock situation occurs around $t = 170\,\mathrm{ms}$ (BTD), although the neuron's analog behavior does not seem to be affected.

4.3.2 Firing Threshold vs. Reset Potential

The hardware parameter $I_{\mathrm{thresh}}^{\mathrm{bias}}$ determines a bias current for the firing threshold comparator circuits in every `FHW-1` neuron (see Section 2.1.4). It controls the responsiveness and, due to its amplifying effect on transistor mismatch, the effective value of the firing threshold. High values of $I_{\mathrm{thresh}}^{\mathrm{bias}}$ minimize the offset caused by transistor mismatch. At the same time, $I_{\mathrm{thresh}}^{\mathrm{bias}}$ controls the length of the period during which, after a spike has been triggered, a strong conductance is enabled that connects the membrane with the reset potential V_{reset}. High values of $I_{\mathrm{thresh}}^{\mathrm{bias}}$ shorten this period of conductivity, low values increase it, i.e. the mechanism determines the effective value of the neuron refractory period τ_{ref} (see Section 2.1.2) or even the effective reset potential in case the time is even too short to let the membrane reach V_{reset}.

This dual function of $I_{\mathrm{thresh}}^{\mathrm{bias}}$ implies conflicting objectives. Since the threshold comparator circuits are subject to transistor level variations, the parameter $I_{\mathrm{thresh}}^{\mathrm{bias}}$ should be used to tune the chip such that all neurons respond similarly to identical input. This optimization turned out to be very important, because if $I_{\mathrm{thresh}}^{\mathrm{bias}}$ is set too low, the fluctuation of the effective firing thresholds due to transistor mismatch can become significant and make certain neurons extremely responsive to synaptic stimulation, while other neurons exposed to the same input do not fire at all. If the distance between the resting potential and the desired firing threshold is configured to be smaller than the threshold fluctuations, certain neurons will start to fire even without any stimulation.

Neurons that are firing at high rates without or with only a marginal stimulation are normally not tolerable, because they distort the network dynamics and consume a lot of the available readout bandwidth, which is already a limiting factor for experimental possibilities (see Section 4.3.7). The opposite effect, i.e. effective firing thresholds that are too high, is also observed and unwanted either, although then the network dynamics and readout bandwidths are usually not affected that strongly.

If the value of $I_{\mathrm{thresh}}^{\mathrm{bias}}$ is set rather high for a neuron in order to cope with one of the two scenarios just described, for example by a calibration routine that tries to counterbalance the unwanted effects, the conductance period pulling the membrane towards its reset potential can become too short. Then the membrane might not have enough time to reach the reset voltage at all, i.e. the effective reset mechanism is too weak. This can even lead to a state where the membrane potential fluctuates around a value close to the firing threshold, while the neuron does not output digital spikes anymore and hence completely loses functionality. For some neurons, this phenomenon has been observed to already occur for $I_{\mathrm{thresh}}^{\mathrm{bias}}$ set to values of e.g. $0.4\,\mu\mathrm{A}$ (`HVD`) while being exposed to strong synaptic stimulation. The Analog Unit Test `AUT_shortReset` provides a setup that reproducibly shows this specific behavior, e.g. on `FHW-1.3-No.18` (see Appendix A.2).

Furthermore, if $I_{\mathrm{thresh}}^{\mathrm{bias}}$ is set to low values, the membrane can be caught in a sort of analog deadlock that makes it start to permanently fluctuate around its reset potential just after the first spike has occurred. For this case, the Analog Unit Test `AUT_longReset` is a reproducible example, e.g. on `FHW-1.3-No.18` (see Appendix A.2).

For multiple neurons on a typical `FHW-1` chip, no value for $I_{\mathrm{thresh}}^{\mathrm{bias}}$ can be found that avoids all of these unwanted extrema. Thus, the `FHW-1` user has to deal either with sets of highly sensitive and sets of insensitive neurons or with neurons that have a very weak effective reset mechanism. A calibration method has been developed that, for every individual neuron, searches the best value for $I_{\mathrm{thresh}}^{\mathrm{bias}}$, with respect to the following three criteria:

- The neuron is not allowed to fire a spike when V_{rest} is set Δv_{low} below V_{thresh} (lower tolerance cut for threshold transistor mismatch) $\Rightarrow$ minimum value for $I_{\text{thresh}}^{\text{bias}}$.

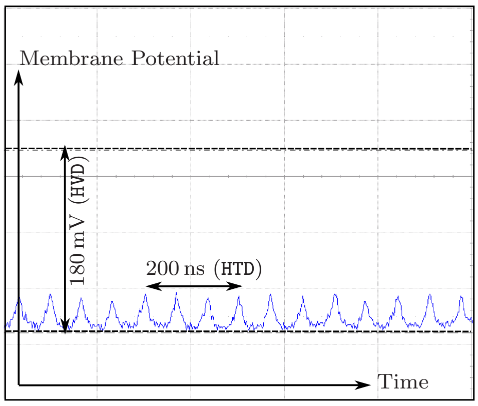
- The neuron has to fire a minimum spike rate when V_{rest} is set Δv_{high} above V_{thresh} (upper tolerance cut for threshold transistor mismatch) $\Rightarrow$ minimum value for $I_{\text{thresh}}^{\text{bias}}$.
- The neuron's effective reset potential is not allowed to be more than Δv_{reset} above the desired value $\Rightarrow$ maximum value for $I_{\text{thresh}}^{\text{bias}}$.

A calibration run on `FHW-1.3-No.17` revealed that for tolerance limits of $\Delta v_{\text{low}} = \Delta v_{\text{high}} = 5\,\text{mV}$ and $\Delta v_{\text{reset}} = 3\,\text{mV}$ (`BVD`), 38 out of the 384 neurons do not satisfy these demands and thus are tagged to be unusable.

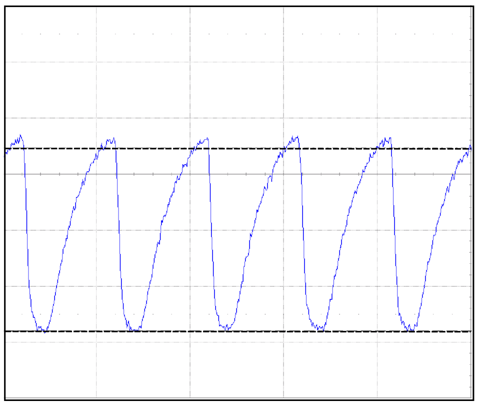
Figure 4.14 shows oscilloscope screen-shots which illustrate most of the possible membrane dynamics problems for different settings of $I_{\text{thresh}}^{\text{bias}}$. It shows a single neuron with its firing threshold V_{thresh} set below the resting potential V_{rest}. Thus, it should fire permanently, resulting in a fast oscillation between V_{thresh} and V_{rest}. For the recording in sub-figure **(a)**, $I_{\text{thresh}}^{\text{bias}}$ is set to $0.0\,\mu\text{A}$ (`HVD`), which is not a reasonable setting, but it shows the possible sticking to V_{reset}, probably due to a nearly permanently active conductance towards this potential. Sub-figure **(b)** shows an acceptable membrane behavior for $I_{\text{thresh}}^{\text{bias}}$ set to $0.01\,\mu\text{A}$ (`HVD`). But, as can be seen in the following pictures, the real firing threshold (upper dashed line) is clearly exceeded before the reset mechanism is triggered, which might be explained by a slow comparator. Still, the reset mechanism seems to manage to pull the membrane fully down to the reset potential. Sub-figures **(c)** and **(d)** with $I_{\text{thresh}}^{\text{bias}}$ set to $0.1\,\mu\text{A}$ and $0.3\,\mu\text{A}$ (`HVD`) illustrate the shortening period of active conductance towards V_{reset}, as the membrane cannot reach the reset potential anymore. In sub-figure **(e)** ($I_{\text{thresh}}^{\text{bias}} = 0.4\,\mu\text{A}$, `HVD`), the neuron already fluctuates around the threshold, but still generates spikes (not shown in the plot), while in sub-figure **(f)** ($I_{\text{thresh}}^{\text{bias}} = 0.5\,\mu\text{A}$, `HVD`) the membrane is nothing but a flat line above the threshold, and the neuron does not output a single digital spike signal anymore.

One negative side-effect of these variabilities of effective thresholds and reset potentials is that the application of a purely spike-based and thus fast method, which had been developed to determine membrane time constants (see Section 5.2.3), has become impossible. This method had to be extended such that it now dynamically determines the effective reset, resting and threshold potentials for every individual neuron. For this purpose, the analog sub-threshold information acquired via oscilloscope has to be incorporated into the calibration algorithm, which makes the routine very slow.

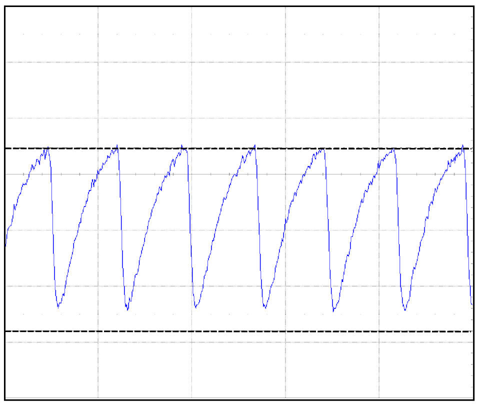
A possible and intended solution is to provide two parameters for the disjoint functionalities of threshold sensitivity and reset efficacy in the planned fourth version of the `FHW-1`. This will uncouple the two mechanisms and allow for an appropriate configuration of each neuron.



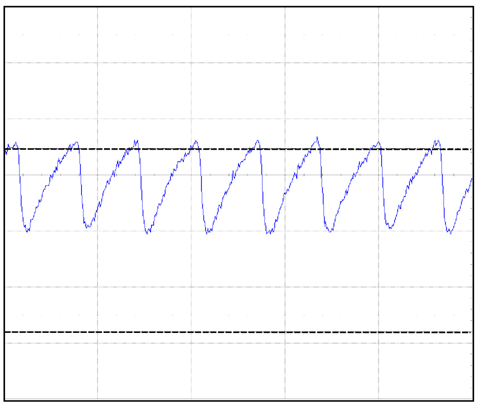
(a) $I_{\mathrm{thresh}}^{\mathrm{bias}} = 0.0\,\mu\mathrm{A}$



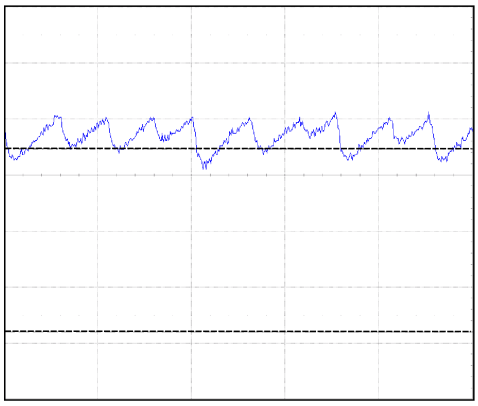
(b) $I_{\mathrm{thresh}}^{\mathrm{bias}} = 0.01\,\mu\mathrm{A}$



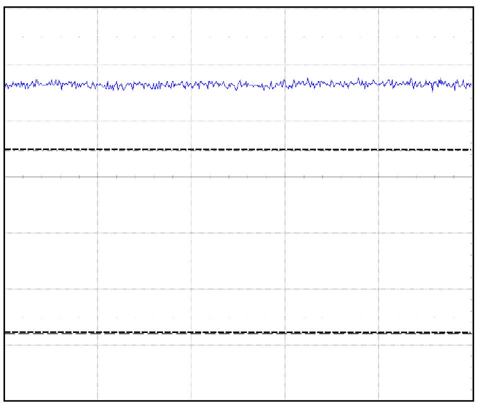
(c) $I_{\mathrm{thresh}}^{\mathrm{bias}} = 0.1\,\mu\mathrm{A}$



(d) $I_{\mathrm{thresh}}^{\mathrm{bias}} = 0.3\,\mu\mathrm{A}$



(e) $I_{\mathrm{thresh}}^{\mathrm{bias}} = 0.4\,\mu\mathrm{A}$



(f) $I_{\mathrm{thresh}}^{\mathrm{bias}} = 0.5\,\mu\mathrm{A}$

Figure 4.14: The membrane potential of an `FHW-1` neuron that should permanently fire since its firing threshold is set to a value below its resting potential. The sub-figures show recordings from runs with identical settings except of the value for parameter $I_{\mathrm{thresh}}^{\mathrm{bias}}$. The choice of this value obviously affects the membrane dynamics strongly. For a detailed description of the observable phenomena see text.

4.3.3 Parasitic Resting Potential Offsets

It has been observed that connecting synapse drivers to a neuron membrane, even without applying any spike stimulation, causes a shift in the resting potential of the neuron. Connecting excitatory synapse drivers increases and connecting inhibitory drivers decreases the resting potential. The Analog Unit Test `AUT_synDriverLeakage` provides a reproducible setup that shows this phenomenon, e.g. on `FHW-1.3-No.18` (see Appendix A.2). When varying the number of drivers or the weights with which these drivers are connected, the effect on the resting potential varies correspondingly. Zero weights have no effect.

Figure 4.15 shows measurements acquired from `FHW-1.3-No.18` that illustrate the extent of the problem. For a completely unstimulated neuron on `FHW-1.3-No.18`, the average membrane potential is measured as a function of the number and weights of the excitatory synapse drivers connected to it. The firing mechanism of the neuron is deactivated. The four sub-figures show the same measurements for four different values of the parameter $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ (see Section 2.1.4), the reason of which will become clear further below. The color range of the plots is selected such that black denotes the desired average membrane potential, i.e. the configured value of V_{rest}, here being $-71\,\mathrm{mV}$ (`BVD`). The upper cut-off color white is at a typical value for the firing threshold, here $V_{\mathrm{thresh}} = -55\,\mathrm{mV}$ (`BVD`).

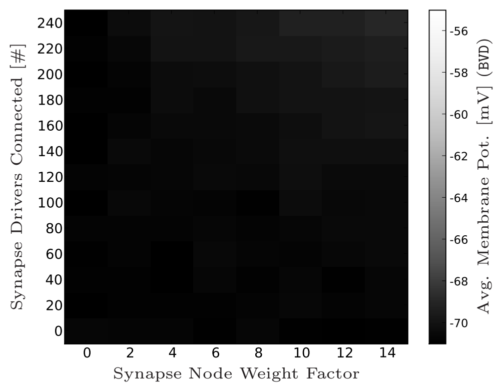
In sub-figure **(a)**, the average membrane potential does not seem to be affected by the connected synapse drivers – it stays around $-71\,\mathrm{mV}$ (`BVD`) for every tested configuration. But in sub-figures **(b)**, **(c)** and **(d)**, a dependence of the unwanted effective resting potential offset on both the number and the weights of the connected synapse drivers becomes obvious. For a large input number and large weights, the offset can easily raise the resting potential above the firing threshold, despite the total absence of stimulation. The four different sub-figures indicate a further parameter dependence, which gives an important hint for the explanation of the problem: The parameter $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ obviously has a huge impact on the magnitude of the offset.

Further investigations revealed that the base lines of the voltage ramps generated at each synapse driver (see Section 2.1.2), in an ideal system determined by the parameter $V_{\mathrm{synbias}}^{\mathrm{ctrl}}$ (see Section 2.1.4), are parasitically raised by a current on the line that connects this externally provided voltage with the drivers. The source of this problematic additional current could not yet be clarified, but it was found to be dependent on the programmable current $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$, which explains the influence of this parameter. A certain current flow via the critical resistors (between the DAC[5] providing $V_{\mathrm{synbias}}^{\mathrm{ctrl}}$ and the output line of the synapse drivers) caused by $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ had been expected from the design, but the amount measured in the laboratory exceeds the expected values by an order of magnitude.

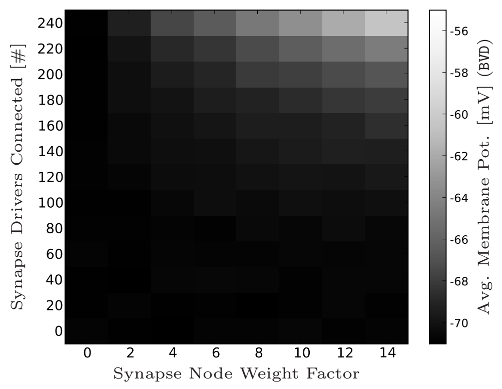
The resting potential offset issue raises major problems: First, as indicated by Figure 4.15, the parameter $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ should be chosen as small as possible in order to minimize the effect of connected synapse drivers. A value of $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}} \leq 0.15\,\mu\mathrm{A}$ (`HVD`) seems to be a good advice. As a further option, Section 5.1.2 describes a possible but rather intricate soldering workaround that can decrease the voltage offset inside the drivers. If this is not applicable or practical, which is usually the case, the only option is to limit $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$, which can mean a loss of control over the synaptic time constants. If these need to be manipulated, especially if they shall be kept rather short (needs large values of $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$[6]), then a setup-dependent distortion of the resting

[5]Digital-to-Analog Converter

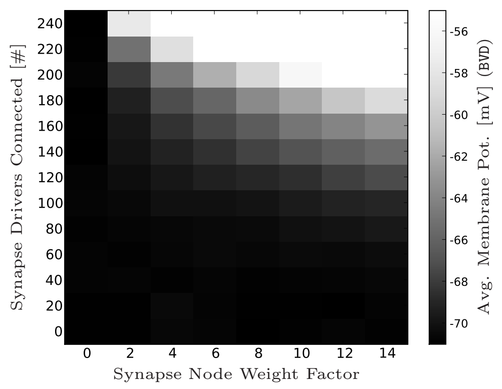
[6]For the `FHW-1.3` chip, the full range of accessible synaptic time constants can be achieved with values of $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ smaller than $0.15\,\mu\mathrm{A}$, but this results from the unwanted fact that the synaptic time constants already



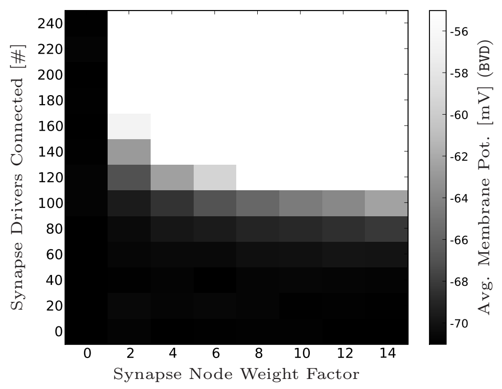
(a) $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}} = 0.15\,\mu\mathrm{A}$



(b) $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}} = 0.30\,\mu\mathrm{A}$



(c) $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}} = 0.50\,\mu\mathrm{A}$



(d) $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}} = 1.00\,\mu\mathrm{A}$

Figure 4.15: Connecting synapse drivers to a membrane imposes a resting potential offset. For different values of the parameter $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}}$ ((**a**) $0.15\,\mu\mathrm{A}$, (**b**) $0.3\,\mu\mathrm{A}$, (**c**) $0.5\,\mu\mathrm{A}$, (**d**) $1.0\,\mu\mathrm{A}$, `HVD`), the average membrane potential (`BVD`) of an unstimulated neuron is plotted as a function of the number of connected excitatory synapse drivers and of the hardware weight factor (possible values between 0 and 15) with which these are connected. Figure by J. Bill.

potential will occur. For a fixed and known setup, such distortions can be counterbalanced by lowering the value of V_{rest}, but this requires a lot of additional, setup-specific calibration effort, and as soon as STDP dynamically changes the synaptic weights in a network, such a pre-calibration would lose its validity. Consequently, in practice there is currently no other option than keeping the parameter $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}}$ in the order of $0.15\,\mu\mathrm{A}$ (`HVD`).

saturate at values much larger than expected. This limitation is described in Section 4.3.5.

4.3.4 Synapse Driver Efficacies and Time Constants

The efficacies of the synapse drivers on an `FHW-1` chip vary strongly, both spatially and, due to an activity dependency, for excitatory drivers also temporally. The spatial effect, i.e. the fact that both PSP[7] amplitudes and time constants are different from actuating synapse driver to synapse driver despite identical settings, is caused by hardware variations on the transistor level and possibly by parasitic capacitances that vary due to locally different surrounding circuitry. This is to be expected and can be counterbalanced, at least to a usually sufficient degree, by calibration routines – an elaborate one is described in Section 5.2.4. In contrast to this, the activity-dependent effect has to be considered a serious issue that makes the efficacy of a synapse driver unpredictable.

A minor problem is the fact that if the current that controls the rising ramp of the conductance courses generated at each synapse driver, $I_{\tau_\mathrm{rise}}^\mathrm{ctrl}$, is chosen too high, the circuit that compares the current voltage with the maximum value (determined by $I_\mathrm{amp}^\mathrm{ctrl}$) and that triggers the change from increase to decrease may be too slow to detect the threshold crossing within a sufficiently short time. This results in a large influence of $I_{\tau_\mathrm{rise}}^\mathrm{ctrl}$ on the resulting voltage ramp amplitude. A high value of the comparator bias current $I_\mathrm{syn}^\mathrm{bias}$ improves its speed and thus minimizes this problem (see Section 2.1.4). Furthermore, since the conductance rise time is assumed to be infinitesimally short in the reference models utilized throughout this thesis, the technique chosen for all later calibration work is to keep the value of $I_{\tau_\mathrm{rise}}^\mathrm{ctrl}$ constant at a large value. Hence, synapse driver calibration methods as the one described in Section 5.2.4 will implicitly consider and counterbalance such overshoots.

The spatial fluctuation of conductance course amplitudes and time constants generated at synapse drivers that are caused by transistor level variations are a fixed-pattern problem which in principle can be calibrated utilizing the parameters $I_\mathrm{amp}^\mathrm{ctrl}$ (for the amplitudes) and $I_{\tau_\mathrm{fall}}^\mathrm{ctrl}$ (for the decay time constant). Figures 4.16 and 4.17 show the different PSP integrals and time constants as determined via the STA method (see Section 4.1.1) for the left block of the `FHW-1.3-No.18` system. For these measurements, all synapse drivers were configured with the values $I_\mathrm{amp}^\mathrm{ctrl} = 0.2\,\mu\mathrm{A}$ and $I_{\tau_\mathrm{fall}}^\mathrm{ctrl} = 0.15\,\mu\mathrm{A}$ (`HVD`). In order to establish a realistic operating point, the tested neuron has been exposed to 208 independent excitatory and 48 inhibitory Poisson spike trains with a firing rate of 3.0 Hz (`BTD`) each. Hence, the PSP integrals are partly positive (index 0 up to 207), partly negative (index 208 up to 255). Each data point in these two figures represents the mean over 10000 individual PSPs, acquired at an average sub-threshold operating point which was actively adjusted. For the excitatory drivers, the mean of the integral distribution is $28 \cdot 10^{-6}$ Vs, the standard deviation is $15 \cdot 10^{-6}$ Vs (`BTD` and `BVD`). The mean of the decay time constant distribution is 34 ms, the standard deviation is 12 ms (`BTD`).

A minimum requirement for any synapse driver calibration would be that the integral over a PSP caused by such a driver (assuming a well defined average membrane potential) does not vary more than a defined value from synapse driver to synapse driver. A better calibration routine would consider both the integral and the decay time constant. Such a dual optimization approach has been developed and is described in Section 5.2.4. But, due to the critical analog effect described above in Section 4.3.3, $I_{\tau_\mathrm{fall}}^\mathrm{ctrl}$ has to be kept as low as possible. Hence, for the available `FHW-1` systems, the decay time constants of synaptic conductance courses cannot be calibrated sufficiently at the moment.

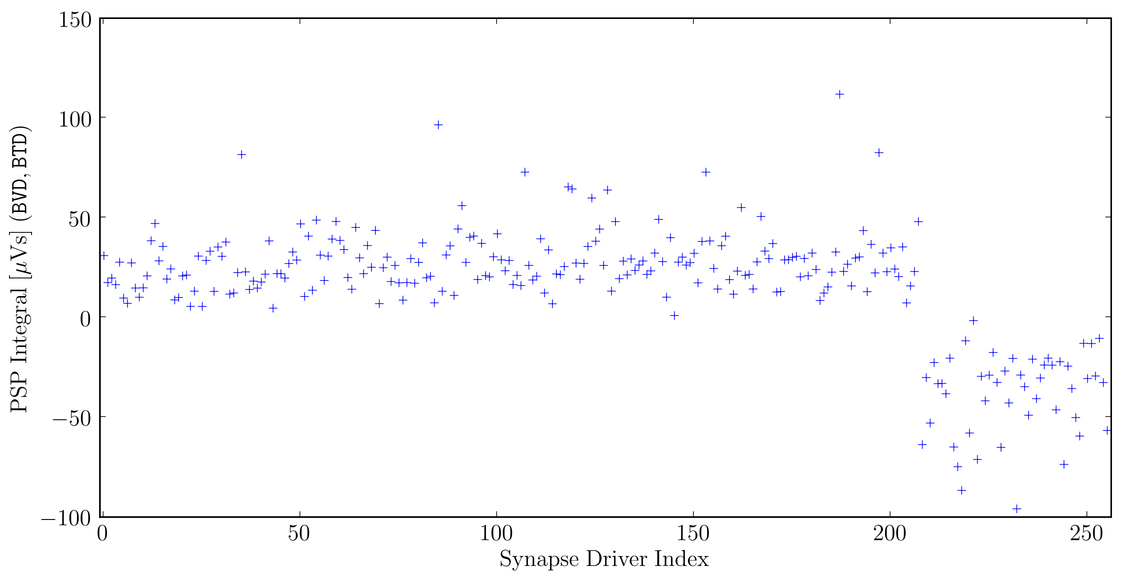
The activity-dependent fluctuations are most probably caused by voltage drops of the

[7] Post-Synaptic Potential

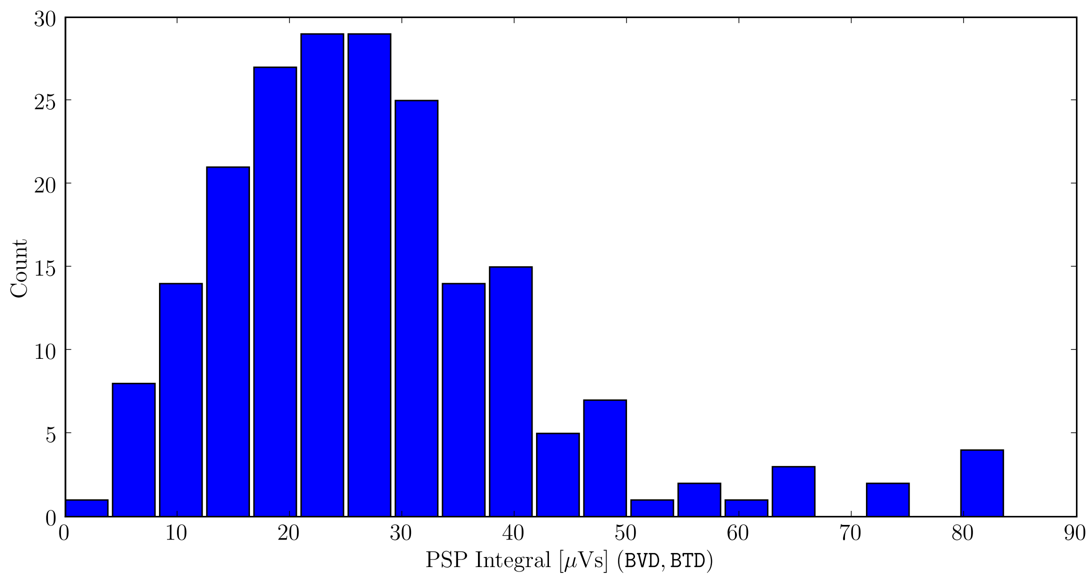
excitatory reversal potential E_e due to too high load, i.e. when too many synapses connect this potential to leaky membranes, it will break down. This cannot be avoided by increasing the bias current $I^\mathrm{bias}_{E_\mathrm{e}}$ for the corresponding hardware voltage generator.

Figure 4.18 illustrates the effect: A single neuron is stimulated by 5 synchronous spikes via 5 synapse drivers, connected with a maximum digital weight to the membrane. For different values of $I^\mathrm{bias}_{E_\mathrm{e}}$, the PSP amplitude is measured as a function of the parameter $I^\mathrm{ctrl}_\mathrm{amp}$, which determines the amplitude of the conductance course generated by the synapse drivers. In the left sub-figure, the plotted neuron is the only one that gets connected to E_e, and only for the said 5 spikes. In the right sub-figure, the same setup is repeated, but a second neuron receives 160 excitatory and 40 inhibitory independent Poisson spike trains firing at a rate of 1 Hz (`BTD`). Obviously, the additional load caused by the 160 excitatory spike trains connecting the second neuron to E_e massively decreases the PSP amplitude of the first neuron, although there is no intended connection or correlation between the two membranes.

The more neurons impose load onto E_e, the stronger this decrease becomes. Hence, the efficacy of a post-synaptic conductance course is strongly dependent on excitatory network activity. For the current system, there is no known way to avoid this issue. The Analog Unit Test `AUT_excLoad` provides a reproducible setup that shows the phenomenon on `FHW-1.3-No.18` (see Appendix A.2). The consequence of these load-dependent weakenings of the excitatory reversal potential is the impossibility to adjust excitatory synapses adequately to a deterministic strength. This fact has to be considered as fatal for a wide field of applications that otherwise might have been possible for the `FHW-1` system.

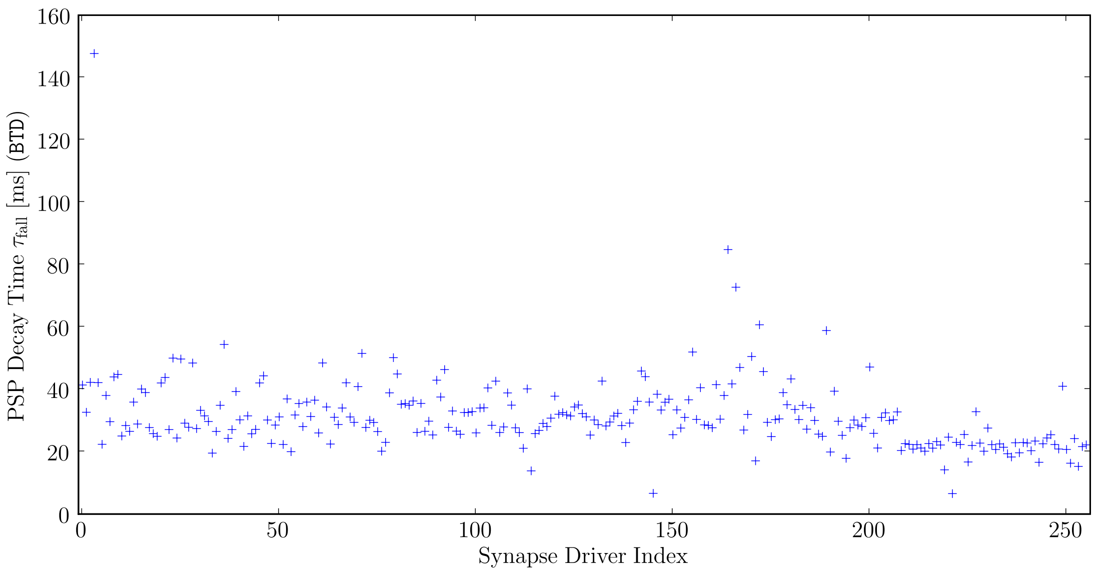


(a) PSP integrals vs. Synapse driver index

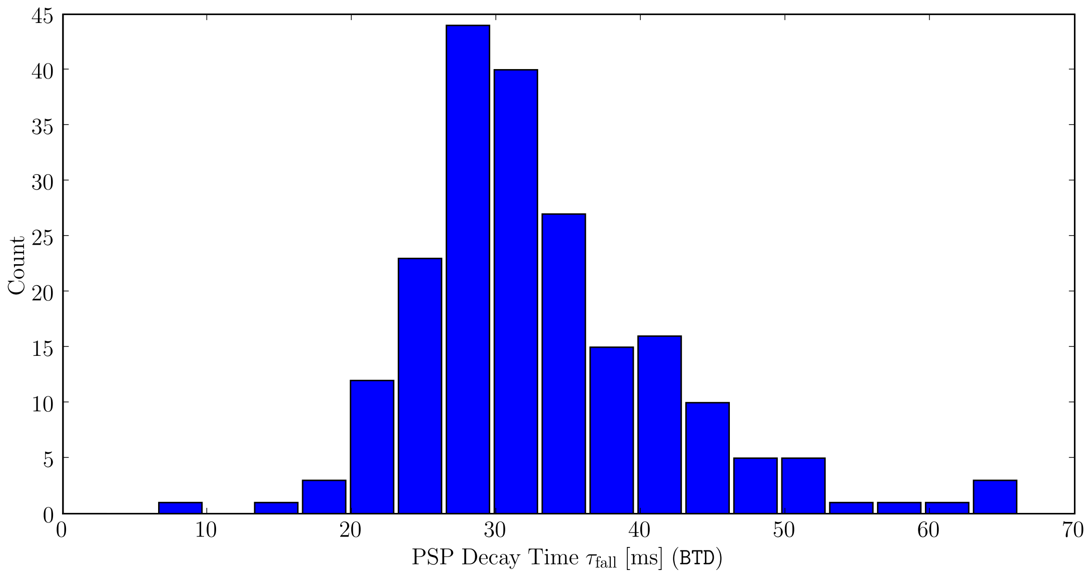


(b) Histogram of PSP integrals (excitatory synapse drivers only)

Figure 4.16: Identically configured synapse drivers result in different PSP integrals. Figure **(a)** shows the measured integrals caused by each of the 256 synapse drivers of the `FHW-1.3-No.18` system's left block. Figure **(b)** summarizes these data points into a histogram.

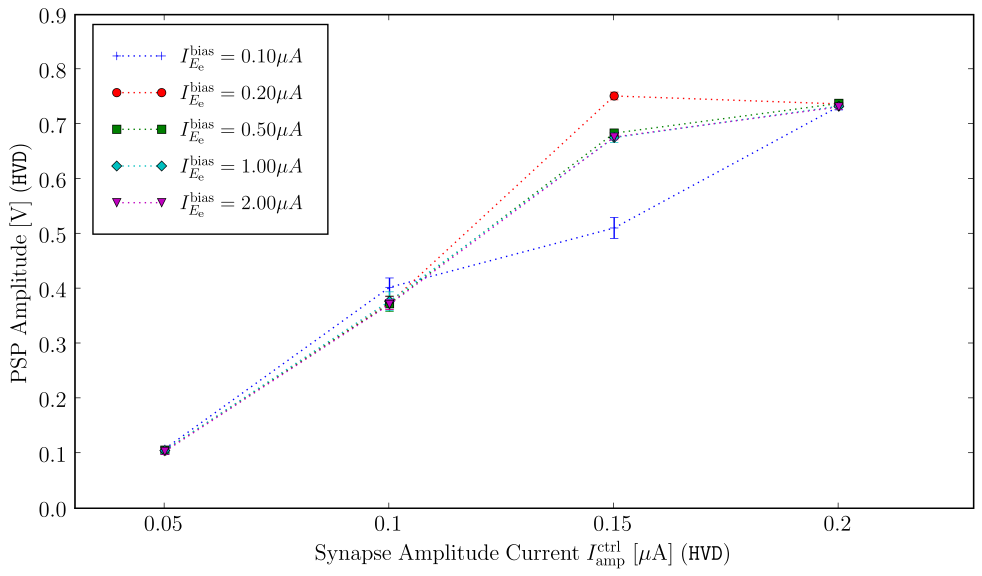


(a) PSP decay time constant vs. Synapse driver index

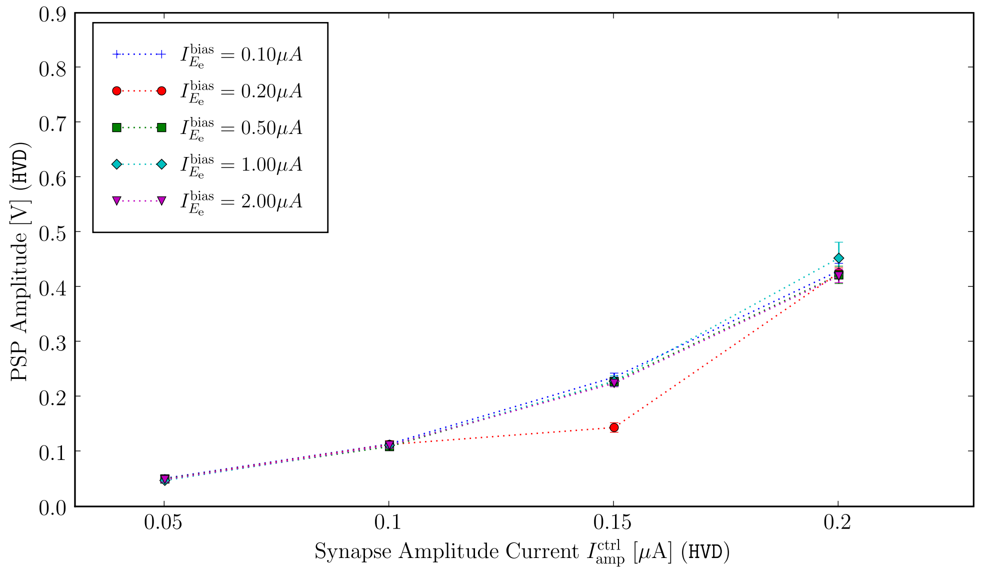


(b) Histogram of PSP decay time constants

Figure 4.17: Identically configured synapse drivers result in different PSP decay time constants. Figure **(a)** shows the measured time constants caused by each of the 256 synapse drivers of the `FHW-1.3-No.18` system's left block. Figure **(b)** summarizes these data points into a histogram.



(a) Low load on excitatory reversal potential E_e



(b) High load on excitatory reversal potential E_e

Figure 4.18: A high load on the voltage generators for E_e causes an activity-dependent decrease of the corresponding synapse efficacies. The plot shows the amplitude of five accumulated, synchronous excitatory PSPs versus the synapse amplitude current I_{amp}^{ctrl}. The chosen range for this control current covers typical values used for the `FHW-1` operation. Every data point represents the mean over 10 runs, the error-bars are the standard deviations of the individual PSPs. In every sub-figure, the curves for five different value settings of the bias current $I_{E_e}^{bias}$ are shown. In sub-figure **(a)**, the load on E_e caused by background activity is very low, while in **(b)** it is much higher. The influence of the load is obvious, while large values of the parameter $I_{E_e}^{bias}$ seem to have no positive effect on this issue.

4.3.5 Dis-Proportionality of Intrinsic Time Constants

In computational neuroscience, typical membrane time constants τ_m for conductance-based I&F neuron models are found between 10 ms and 20 ms (`BTD`) (Kumar et al., 2008; Brette and Gerstner, 2005; Sussillo et al., 2007; Shelley et al., 2002), while synaptic decay time constants usually cover a much wider range. Typical values for excitatory synaptic conductance decays $\tau_\mathrm{syn,E}$ are chosen between 0.3 ms (Kumar et al., 2008) and 4 ms (`BTD`) (Sussillo et al., 2007; Shelley et al., 2002). Inhibitory synapses usually are configured to be a factor 2 to 4 slower.

For the `FHW-1` system, the synaptic decay time constants were originally assumed to be configurable within a range of 10 ns to 1 µs (`HTD`), i.e. covering two orders of magnitude. Figure 4.19 shows the synaptic decay time constant as a function of the parameter $I^\mathrm{ctrl}_{\tau_\mathrm{fall}}$ (which normally should not be chosen larger than 0.15 µA (`HVD`), see Section 4.3.4). All values of τ_syn have been acquired with the STA method (see Section 4.1.1) and a setup which guarantees that the utilized membrane is in a high-conductance state (see Sections 1.3.1 and 4.1.2, see Kaplan, 2008). In this plot, the available time constants range from around 30 ms to varying maxima between 55 ms and 140 ms (`BTD`). The lower limit is typical for all neurons on all chips. As discussed above, this value is much larger than the time constants usually applied in models.

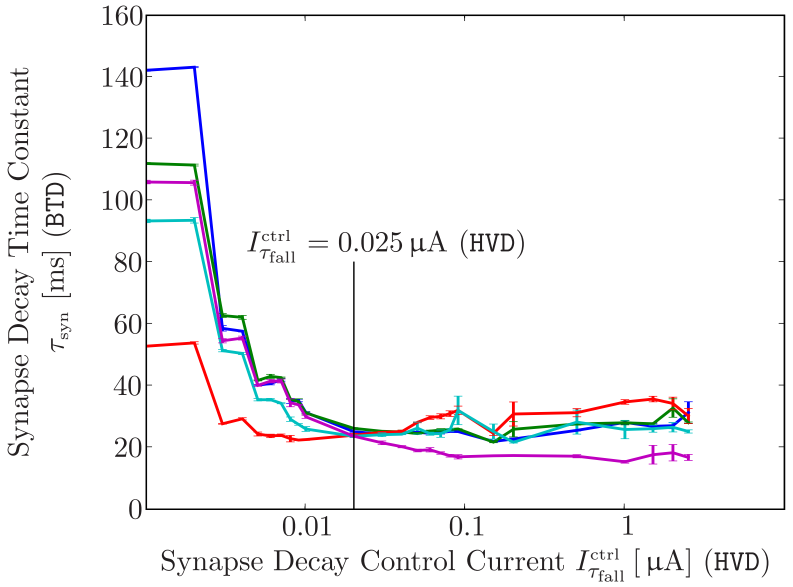


Figure 4.19: Synaptic decay time constants vs. the parameter $I^\mathrm{ctrl}_{\tau_\mathrm{fall}}$ (`HVD`). Data points have been acquired with the STA technique on a membrane in a high-conductance state. For $I^\mathrm{ctrl}_{\tau_\mathrm{fall}} = 0.025$ µA, the difference between the resulting time constants is minimal. Figure by B. Kaplan.

So why not decrease the speedup interpretation by a factor of 10? The problem is the configurable range of membrane time constants, which is approximately 50 ns to 150 ns (`HTD`). For a speedup factor of 10^4, this would mean 0.5 ms to 1.5 ms (`BTD`), being one order of magnitude too short. Since the activity-dependent integration capability of a membrane, i.e. especially its varying low-pass filtering properties, is assumed to be functionally important, a realistic time constant for an unstimulated membrane is demanded. Hence, the translation factor between `HTD` and `BTD` is kept as 10^5, despite the resulting slow synapses. In the specific case of the FHW-1 chip, which has much less synaptic inputs ($N_{\text{inputs}}^{\text{chip}} = 256$) than usually found for a cortical neuron ($N_{\text{inputs}}^{\text{cortex}} \approx 10^3..10^4$), a higher synaptic time constant can possibly counterbalance this lack of stimuli to achieve a comparable level of excitation.

Still, in order to map more network models onto the hardware substrate, it is desirable to have a wider range of configurable ratios $\frac{\tau_{\text{syn}}}{\tau_{\text{m}}}$ available. Hence, for a future FHW-1 version, the neuron membrane capacitance is planned to be increased by a factor of at least 10. A resulting emulation slowdown would also minimize the I/O bandwidth problems summarized in Section 4.3.7.

4.3.6 Multi-Spikes

Many neurons exhibit a double- or multi-spike phenomenon, i.e. for every spike that is visible on the analog membrane trace, two or more spikes with a distance between 3 ns and 7 ns (HTD) are recorded digitally. For affected neurons, this inter-spike distance varies from spike pair to spike pair, even within one experiment run. The Analog Unit Test AUT_multiSpikes provides a reproducible setup that shows this specific behavior, e.g. on FHW-1.3-No.18 (see Appendix A.2). The intensity of this phenomenon has been found to be influenced by the parameter $I_{\text{thresh}}^{\text{bias}}$.

If two adjacent neurons within one neuron block would always show the same behavior, one could speculate about a problem within the digital registering of the spike, as one spike pulse from within the analog part of the chip might have overlapped two clock cycles of the digital part and thus been registered as two events. But there is clear evidence of neurons within one neuron block that behave differently, so there has to be another explanation.

The multi-spikes can have a distorting influence on, for example, synaptic plasticity, if they are already existent within the analog part of the chip. This still has to be investigated. In any case, they distort the digitally recorded event data in terms of precise spike times and firing rates, plus might affect firing regularity, synchrony and other statistical parameters. Artificially increased firing rates can cause further problems, because the readout bandwidth of the FHW-1 system is very limited, as will be described in the following paragraph.

4.3.7 Limited Spike Input and Output Bandwidth

As discussed in Sections 3.1.5 and 4.3.5, the intrinsic membrane and synapse time constants within the FHW-1 chip determine the interpreted translation factor of 10^5 between HTD and BTD. With this speedup and an applied chip clock of $f_{\text{chip}} = 100\,\text{MHz}$ (which, at least for the FHW-1.1 and FHW-1.2 systems, cannot be significantly increased due to problems with the internal DLL locking, see Section 4.3.9), the input and output bandwidths for digital events are very limited.

There are two main bottlenecks for digital data coming into or going out of the chip. First, the packet transport between the controlling FPGA respectively the so-called *Playback Memory* and the chip itself. Since one packet can carry a maximum of three events, the total maximum input and output rate for a chip clock of $f_{\text{chip}} = 100\,\text{MHz}$ is 300 MEvents/s (HTD). This is further limited by the fact that each packet has to carry events for different synapse driver blocks.

The second bottleneck is given by the depth of the FIFO queues that buffer the event input and output on the chip. Each neuron and synapse driver block has two FIFO buffers, each of which is 64 events deep and operates with the chip clock, but one of the two being shifted by the half of one clock period. This results in an upper limit for the input (output) rate of 200 MEvents/s (`HTD`) per synapse driver block (neuron block), eight (six) of which are located on one chip. Still, an input (output) rate of eight (six) times 200 MEvents/s (`HTD`) might not be feasible because of the packet rate described above. The occurrence of bandwidth problems strongly depends on the configuration of the system and on the temporal and spatial distribution of its activity.

To give an idea of realistic limits, two simple example cases will be introduced, which have been tested experimentally. For a speedup factor of 10^5 and a chip clock of $f_{\text{chip}} = 100\,\text{MHz}$, the maximum rate of 256 independent, externally generated Poisson spike trains stimulating one single neuron is approximately 8 Hz (`BTD`) per train. This upper limit arises from the packet transmission limit of 300 MEvents/s (`HTD`). If only 64 external inputs are applied, but to the same synapse driver block (which is a very realistic situation, because 64 adjacent synapse drivers per chip half are not occupied by possible feedback connections), the maximum rate for a Poisson spike train is approximately 11 Hz (`BTD`) per train, in this case determined by the FIFO buffer depth. Both bandwidth limitations are demonstrated by single-neuron experiments presented in Section 6.1.2.

A decrease of the interpreted speedup by a factor s via a redesign of the electronic membrane properties, as proposed in Section 4.3.5, would increase the available biological input and output rates by the same factor s.

4.3.8 Crosstalk of Digital Activity

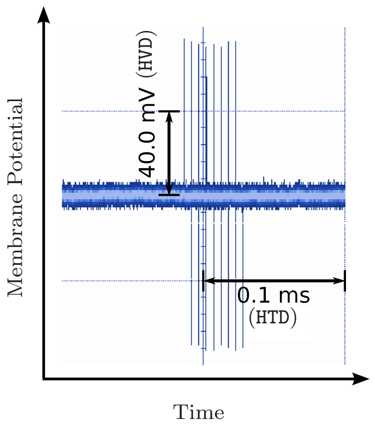
Activity within some circuits of the digital part of the chip causes noise in the purely analog circuitry. This reproducible, clock-dependent noise normally is superposed by hardware-intrinsic, statistical noise (see Section 4.2.2). Still, in some cases this digital crosstalk can become large enough to be visible even on a noisy membrane potential. And if the STA technique (see Section 4.1.1) is applied, i.e. if statistical noise is filtered out, crosstalk effects can become very noticeable.

Figure 4.20**(a)** shows an `FHW-1.3` resting potential during a parameter update of so-called membrane voltage output buffer bias currents. Nine digital spikes can be observed, caused by crosstalk from the update circuits. In this case, the spiking behavior was not observed to be affected, thus this phenomenon is assumed to affect the readout chain, but not the membrane itself.

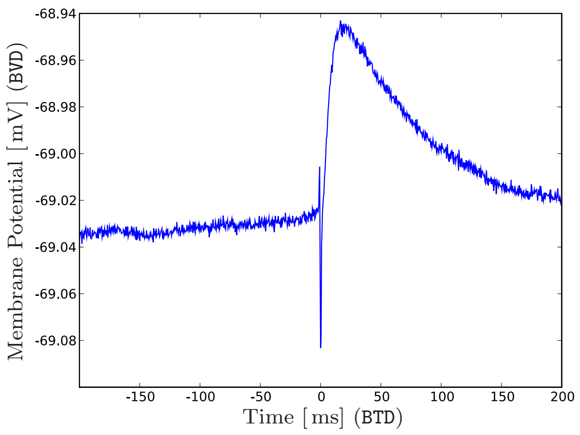
In Figure 4.20**(b)**, an EPSP extracted via STA is shown. A peak is visible just before the EPSP. This is caused by crosstalk from digital circuit activity upon the arrival of the digital spike information, hence its temporal correlation to the PSP is always the same and thus its effect cannot be filtered out by averaging techniques.

Especially the kind of crosstalk shown in Figure 4.20**(a)** can cause a serious problem during calibration of the chip. Although it might not affect the membrane itself, but just the recording of it, this can be a problematic issue in cases where analog readout information is essential, as e.g. for a lot of calibration methods. Determining effective resting and reset potentials or thresholds (necessary for example due to the problems stated in Section 4.3.2) typically involves measuring the minima and maxima of the traces acquired via an oscilloscope. If this information is distorted by digital crosstalk as shown, the results might be misleading.

Future hardware designs might consider more shielding mechanisms for the analog circuitry. Until such a revised system is available, averaging techniques have to filter out at least the



(a) Oscilloscope screen-shot: Crosstalk on resting membrane potential due to parameter updates of the nine *membrane voltage output buffer* bias currents.



(b) A digital crosstalk spike just before the onset of an excitatory PSP (spike-triggered average over 100000 runs).

Figure 4.20: Examples of crosstalk on membrane potentials of `FHW-1.3` neurons.

temporally uncorrelated parts of digital crosstalk.

4.3.9 Clock Problems

For the event data transport to and from the `FHW-1` chips, the periodic signal that clocks the devices, typically oscillating with a frequency of $f_{\text{chip}} = 100\,\text{MHz}$, is first divided into a chip-internal clock of $f_{\text{int}} = 2 \cdot f_{\text{chip}}$. Then this chip-internal clock is divided again into 16 so-called *time bins* by a delay-locked loop (DLL), an electronic building block on the chip. Event delivery to and from the chip needs precise time stamps for each event and hence makes use of this sub-clock. Two problems with the DLL cause situations where events within the last few of each set of 16 time bins are not generated in the analog part of the chip or are dropped during the time-to-digital conversion phase.

One possibility is that the DLL does not synchronize correctly to the chip clock, i.e. that there are either too few or too many time bins per clock cycle. This issue is explained in detail in (Grübl, 2007, Section 6.5). It can lead to a systematic event loss during both stimulation and recording. For $f_{\text{chip}} = 100\,\text{MHz}$, though, the event loss rate due to DLL locking problems was found to be below 0.5% (Grübl, 2008).

The other kind of possible input spike loss is due to too short trigger signals sent to the synapse drivers, so that the conductance course sequence (see Section 2.1.2) will not be started. This problem is experimentally shown and explained in more detailed in Section 6.1.1, where the precision of spike delivery is analyzed with an `FHW-1.2` chip. A workaround that avoids stimulation distortions due to these too short synapse trigger signals is presented in Section 5.1.3. A modification in the circuitry of the third version of the `FHW-1` chip addresses this problem, but has not yet been evaluated.

4.3.10 Insufficient Parameter Range for Synaptic Facilitation and Depression

In order to get short term depression and facilitation working in a biologically realistic regime, the reference voltages V_{fac} and V_{max} (see Sections 2.1.3 and 2.1.4) have to be adjustable within a large range. Otherwise, the control over the impact of the electronically modeled inactive partition (see Section 2.1.3) is very limited, which results in extremely fast saturation at the maximum or minimum synaptic weights.

The voltage generators that are located on the chip (see Section 2.1.4) usually provide values from around 0.6 V to 1.6 V (HVD). For a biologically realistic configuration of the short-term synaptic plasticity mechanism, this has been found to be not sufficient (Bill, 2008). In Section 5.1.4, a workaround is presented which manages to pull down selectable reference voltages like V_{fac} significantly by connecting it to an external potential. This method imposes a high risk of damaging the circuits by wrong usage, and the actual value of V_{fac} is unknown during its application. Furthermore, the method occupies I/O resource that might be needed otherwise, and it requires additional devices like external voltage generators.

4.3.11 STDP Control Problems

The hardware STDP mechanism, which is explained in Section 2.1.3, has not yet been put under satisfying control. Despite considerable efforts (see description of applied technique in Section 4.1.3), no method could be established so far which allows to configure the chip such that all synapses exhibit acceptable weight change curves. Hence, to which extent synaptic long-term plasticity does or does not work is not yet clear. But the measurements indicate that it will be impossible to reliably tune the ratio of the integrals over the causal and the acausal branch of STDP curves with a precision in the order of 5%. Hence, unsupervised STDP learning paradigms that rely on such a precise balancing (see e.g. Song et al., 2000) are not applicable.

Possible reasons for the continuous lack of STDP configurability are the following:

- Since the applied measurement method is rather complex, systematic errors within the high-level routines cannot be ruled out perfectly.
- The results might be distorted by leakage currents inside the correlation measurement circuits that vary from synapse to synapse and over time.
- The low-level software interface does not provide a cleanly encapsulated and fully tested STDP configuration interface – thus there might be errors in its functionality or usage.
- Due to the problems with the effective firing threshold (see Section 4.3.2), it is hardly possible to make a whole set of neurons reliably generate the required post-synaptic spikes. And since each neuron has to be tuned individually, correlation measurements of large synapse arrays become very slow.
- Due to slow measurement methods, possible temperature dependencies of leakage currents discharging the involved capacitors might have a significant influence.
- Fundamental misbehavior of the hardware itself.

Exemplary results that illustrate the mentioned difficulties are presented in Section 6.1.5.

Weight Update Cycle

A well-known issue is the possibly very slow weight update cycle of the hardware STDP mechanism. The weight updating works sequentially, therefore an update is delayed until the controller processes the particular synapse. The following formula can be used to calculate the worst case delay for updating the synaptic weights, assuming that all synapses are plastic, that all synapse rows are used and that all weights need to be updated:

$$t_{update} = (2N_{\mathrm{updates}/(6\times\mathrm{row})} + t_{\mathrm{row\ delay}}) \cdot 2t_{clk} \cdot N_{\mathrm{rows}} \qquad . \tag{4.6}$$

Every two clock cycles up to six weights can be updates. Each row access must be done twice, once for causal and once for acausal measurements. Using the default operating parameters at the time of writing, row access $t_{\mathrm{row\ delay}}$ requires typically 50 cycles; t_{clk} is set to 200 MHz (`HTD`). Thus $(2 \cdot 64 + 50) \cdot 2 \cdot 5\,\mathrm{ns} \cdot 256 \approx 456\,\mu\mathrm{s}$ (`HTD`) are needed for updating all $192 \cdot 256$ synapses within a block. Considering a speedup of 10^5, this represents a worst case of 45.6 s in `BTD`. In a typical setup, values between 10 s and 30 s are expected. This clearly limits the speed of synaptic weight adaptation via STDP in the hardware system. Therefore, depending on the experimental setup, a sufficiently fast weight modification behavior that reproduces existing STDP modeling approaches (reviewed in Morrison et al., 2008) cannot be guaranteed.

4.3.12 Spontaneous Ghost Events

When output spikes are read back from the `FHW-1` playback memory, often digital events are among the data that have no analog correspondence, i.e. that are generated wrongly.

The Analog Unit Test `AUT_ghostSpikes` provides a reproducible setup that shows this specific behavior, e.g. on `FHW-1.3-No.18` (see Appendix A.2): There, one neuron is created and its membrane is recorded for 100 ms (`BTD`) while no stimulation is applied. No single neuron is flagged to be recorded digitally. During the experiment, the membrane remains at its resting potential, but after the run is finished, multiple spikes can be read back from the playback memory. These ghost events seem to occur randomly in time, but from specific neurons only, independent of the neuron the membrane or spikes of which is recorded. For the `FHW-1.3-No.18` system, the neuron indices of ghost events usually are in $I = \{1, 64, 128, 320\}$, i.e. indices close to boundaries of neuron blocks seem to be preferred. The membranes of these neurons that are wrongly indicated to be active show no significant deviation from their resting potentials either.

It is not clear at what point in the event generation chain those ghost events emerge, and due to the fact that such events are rather rare, debugging this issue has been assigned a low priority (Schemmel, 2008).

5 Establishing Biologically Realistic Regimes

After practical difficulties with the operation of the accelerated FACETS hardware system (FHW-1) as a neuroscientific modeling tool have been identified in the previous chapter, the following sections describe methods that help to overcome the hardware-specific drawbacks. The presented techniques allow to set up biologically realistic regimes of activity on the neuromorphic system despite the imperfections of the substrate. Section 5.1 presents techniques that minimize or avoid the corruptive impact of some of the design-related hardware malfunctions described in Section 4.3. The calibration routines presented in Section 5.2 provide a parameter fine-tuning that increases the homogeneity of the neuromorphic sub-circuit dynamics. This is necessary due to process-inherent transistor-level fluctuations and further design-related interferences.
In order to evaluate the success of hardware tuning methods in terms of a possibly increased biological relevance, comparison paradigms with established software simulators are helpful. For this purpose, descriptors of network dynamics are motivated and introduced in Section 5.3. They can be used to quantify the similarity of simulated and emulated data.

5.1 Handling of Chip Imperfections

The following techniques have been developed in order to minimize or avoid the impact of some of the problems with the FHW-1 system described in Section 4.3.

5.1.1 Releasing Recording Deadlocks

The event readout deadlocks described in Section 4.3.1 have been found to be resolvable during an experiment by applying a specific mechanism called *neuron reset*. A neuron reset is a chip-global signal which disables the spiking mechanism while it resets the priority encoders in the event acquisition circuitry of the FHW-1 chip. Those encoders are assumed to cause the deadlocks. After applying a neuron reset signal, events from all neurons can be read again, but only until the next deadlock occurs. Hence, if more than a small subset of the network shall be recorded at the same time and continuously, neuron resets have to be applied many times during an experiment.

Although such a reset signal does not affect the analog integration process of the neurons, the fact that no spike mechanism can be initiated during a reset distorts the network dynamics in various ways. Depending on the length of the neuron reset signals and the frequency with which they are applied, output spikes can be lost or delayed, and the membrane potential might spend significantly more time above the firing threshold compared to a situation with an undisturbed spiking mechanism. Nevertheless, since all available versions of the `FHW-1` system are subject to the deadlock issue, the possibility of recording the whole chip at the same time might be worth the drawbacks caused by the neuron resets.

Different ways of applying neuron resets into an active network have been considered, including periodic resets, resets as Poisson processes, or resets which are applied every nth externally generated input spike. Any method which is based on the external stimulation of the network has been skipped, because self-stimulating networks, especially networks with self-sustaining activity, can run into deadlocks without a single externally applied spike. Regular and Poisson process resetting has been implemented (Müller, 2008, Section 4.3.2). Figure 5.1 shows two neurons which, due to their firing threshold being set below their resting potential, fire permanently at very high rates. But as can be seen from the plot, they run into deadlocks multiple times. By applying multiple neuron resets which last less than 1 ms (`BTD`) each, the deadlocks are resolved for certain periods again. In some cases only one of the neurons is unlocked, in most cases both are.

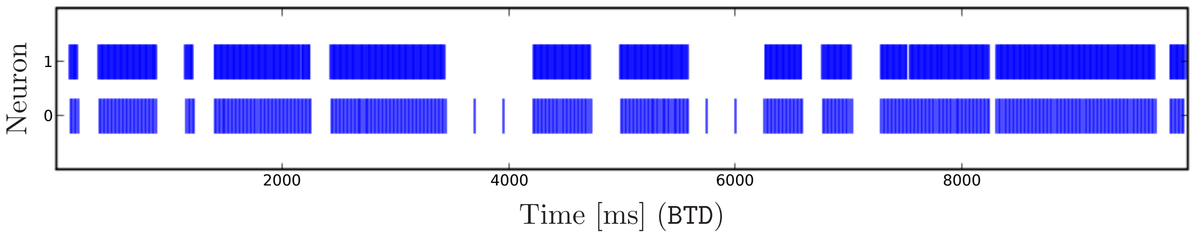


Figure 5.1: The spike output of two neurons which repeatedly run into recording deadlocks. The deadlocks are resolved by so-called neuron resets several times. Time axis in `BTD`. If not caught in a deadlock situation, the firing rates of both neurons are very high in order to provoke the problem. Figure by E. Müller.

5.1.2 Clamping Synapse Driver Base Lines

For suppression of the synapse driver offset voltage as described in Section 4.3.3 and for an applicable synaptic short-term plasticity configuration (Bill, 2008, Section IV.5.4), the parasitically distorted parameter $V_{\mathrm{synbias}}^{\mathrm{ctrl}}$ has to be lowered. This can be achieved by bypassing the connection between the DAC on the Recha board and the on-chip $V_{\mathrm{synbias}}^{\mathrm{ctrl}}$, leaving only an internal resistor of $10\,\Omega$ (`HVD`) across which the parasitically injected currents cause the corruptive voltage deviation. Such a bypass needs additional soldering and was performed only for the `FHW-1.3-No.25` system. With that device the short-term synaptic plasticity experiments presented in Section 6.2.2 were performed, which needed a minimized value of $V_{\mathrm{synbias}}^{\mathrm{ctrl}}$.

5.1.3 Avoiding Time Bin Losses

As a workaround for the too short trigger signal issue described in Section 4.3.9, a technique is applied which guarantees the delivery of all generated stimulation data into the analog part of the chip, provided that the delay-locked loop, which generates the 16 time bins per chip clock cycle, works correctly.

With the chip clock f_{chip} set to 100 MHz (`HTD`), only events in the last one or two time bins within one clock cycle have a non-marginal probability to be lost, i.e. the bins with the indices 14 and 15 or, in binary representation, `1110` and `1111`. Hence, in order to avoid events within these time bins without imposing systematic input pattern distortions, the last two bits of all stimulation event time stamps are set to zero. By doing so, the critical time bins are avoided, and the only drawback is an effective loss of temporal resolution by a factor of 4 – which is acceptable at the current stage of development and in the light of more serious malfunctions presented in Section 4.3.

5.1.4 Providing Sufficiently Low Reference Voltages

The dynamic ranges of programmable voltages provided by the on-chip generators (see Section 2.1.4) are sometimes not sufficient. As for e.g. a useful short-term synaptic plasticity configuration (see Section 4.3.10), values significantly smaller than 0.6 V (`HVD`) are required. Hence, a method has been developed which can pull selectable voltage parameter values towards an externally provided potential (Bill, 2008, Section IV.5.4). For this purpose, the $I_{\text{b}}^{\text{test}}$ pin is used (see Section 2.1.5), which was originally designed as a readout pin only, but which can be used to influence selectable voltage parameters with a potential connected to that pin, i.e. to effectively *write* values.

Two conditions have to be met in order to avoid damage to the $I_{\text{b}}^{\text{test}}$ circuits: The bias currents of all voltage generators connected to the $I_{\text{b}}^{\text{test}}$ pin in its *write* mode have to be set to zero, and the same holds for the target values of the involved voltage parameters defined by the software interface. The total current through the $I_{\text{b}}^{\text{test}}$ pin must not exceed 3 mA (`HVD`).

For the `PyNN.hardware.stage1` module (see Section 3.2.2), a flag is implemented which automatically performs the automatable safety precautions, i.e. it sets the bias currents for the appropriate voltage generators to zero, as soon as configurable voltage parameter values are defined to be externally provided. This workaround is applied for the experiments presented in Section 6.2.2.

5.1.5 Achieving Sufficient Parameter Ranges

During the work with `FHW-1` devices, the output range of the on-chip voltage generators (see Section 2.1.4) turned out to have a critical influence on the usability of the system. The available voltages directly determine the available dynamic range of the hardware neuron membranes, while the various noise sources, parameter fluctuations and parasitic effects (see Section 4.3) impose a total error onto every experiment which is mostly independent of that range. Hence, in order to achieve an acceptable signal-to-noise ratio, the voltage ranges have to be maximized.

The upper and especially the lower limits of such a programmable voltage V can be optimized by selecting an appropriately low value for the bias current $I_{\text{V}}^{\text{bias}}$ of the corresponding voltage generator circuit. According to recommendations originally given by the hardware developers, these bias currents were set to 0.2 µA (`HVD`) for all generators, resulting in a lower output limit of approximately 0.85 V (`HVD`). After checking experimentally that bias current values as small as 0.02 µA (`HVD`) still do not cause significant output voltage drops under realistic load, the resulting lower limit of approximately 0.6 V (`HVD`) can now be regarded as a factual minimum.

During the process of bias current testing, another phenomenon has been observed: Both the bias currents and the absolute values of adjacent voltage memories can affect the effective output value of a voltage memory which itself is not changed in terms of its input value or its bias current. It is assumed that this phenomenon is related to the issue described in Section 4.3.3.

5.2 Hardware Calibration

In this section, a set of calibration mechanisms is described that have been developed to increase the reliability of parameter programming for the `FHW-1` system and to minimize the effects of transistor mismatch from neuron to neuron and from synapse to synapse. The goal is to provide a neural substrate with a maximum of homogeneity in terms of its response to external stimulation. Without these calibration procedures, neurons on the same chip can respond such differently to identical input, that one cell fires permanently at biologically unrealistic rates, while the membrane of another cell is not even close to its firing threshold. The impact of a spike arriving at one synapse driver can be orders of magnitude stronger than the impact of a spike arriving at another synapse driver on the same chip. Under these conditions, it is hardly possible to set up reasonable experiments.

Some of the effects counterbalanced by the presented methods are process-inherent and have been expected, i.e. the corresponding calibration approaches can be transferred to any comparable system. Others, which deal with malfunctions described in Section 4.3, will lose relevance as soon as systems are available that solve the responsible issues.

Johannes Bill, Eric Müller and Andreas Grübl have contributed to the development of the methods presented in this section.

Calibration Framework Except of the voltage generator gauging presented in Section 5.2.1, all calibration methods presented in this section have been implemented in Python and utilize PyNN (see Section 3.1.3) for the setup and control of hardware experiments. The membrane time constant calibration (Section 5.2.3), the synapse driver calibration (Section 5.2.4) and the synapse weight calibration (Section 5.2.5) use the software simulator NEST (see Section 3.1.4) as a reference. This comparison paradigm is not just a control mechanism. It is *the* essential mechanism with which the biologically relevant operation regime of the `FHW-1` hardware is established.

The order in which the routines are presented is the order in which they are recommended to be applied to an uncalibrated chip, as each calibration builds upon and requires its predecessors. A top-level calibration framework has been implemented in Python which embeds all routines presented in the following and which provides the storage of the resulting calibration data for later use in experiments. Hence, for a given workstation (see Section 3.2.4), the full calibration program sequence has to be performed only once.

5.2.1 Voltage Generator Calibration

In the `FHW-1` model, important neuron voltage parameters like its firing threshold and its reversal potentials are generated on-chip by programmable voltage generators (see Section 2.1.4). For those, the relation between the written voltage value U_{in} and the actually generated output voltage U_{out} is well approximated by a linear dependency with a slope m_{U}

and an offset U_{off}, which is limited by upper and lower bounds U_{upper} and U_{lower}:

$$U_{\text{out}} = \begin{cases} U_{\text{out}}^{\text{min}} \,, & \text{if } \; U_{\text{in}} < U_{\text{lower}} \\ U_{\text{off}} + m_{\text{U}} \cdot U_{\text{in}} \,, & \text{if } \; U_{\text{lower}} \leq U_{\text{in}} \leq U_{\text{upper}} \\ U_{\text{out}}^{\text{max}} \,, & \text{if } \; U_{\text{upper}} < U_{\text{in}} \end{cases} \quad . \tag{5.1}$$

The values of m_{U}, U_{off}, $U_{\text{out}}^{\text{max}}$ and $U_{\text{out}}^{\text{min}}$ have been found to differ significantly from voltage generator to voltage generator. This can be explained by fluctuations of the $10\,\text{k}\Omega$ polysilicon resistors which are used to convert the currents that are generated by a DAC[1] into the desired voltages. In the utilized chip production process, polysilicon resistors can have a mismatch of up to 30% (Schemmel, 2008).

The effect is too strong to be ignored, hence a calibration routine has been developed by Eric Müller which, for a given chip, automatically determines the relationship between U_{in} and U_{out}, i.e. it finds and stores the values of m_{U}, U_{off}, $U_{\text{out}}^{\text{max}}$ and $U_{\text{out}}^{\text{min}}$ for every single one of the 40 voltage generator on a chip (Müller, 2008, Section 4.3.4). This calibration algorithm has been embedded into the full chip calibration framework presented here. The acquired translation values are utilized every time a voltage is written to the chip, and the upper and lower limits of the programmable voltage range are incorporated into the automated mapping process between biological (`BVD`) and hardware voltage domain (`HVD`) presented in Section 3.1.5.

5.2.2 Firing Threshold and Reset Mechanism Calibration

The problems inherent to the parameter $I_{\text{thresh}}^{\text{bias}}$ have been discussed in detail in Section 4.3.2. As indicated there, for multiple neurons on a typical `FHW-1` chip, no value for $I_{\text{thresh}}^{\text{bias}}$ can be found which avoids all possible malfunctions caused by this parameter. A calibration method has been developed which, for every individual neuron, searches the best value for $I_{\text{thresh}}^{\text{bias}}$, with respect to the following three criteria:

- The neuron is not allowed to fire a spike when its resting potential V_{rest} is set Δv_{low} *below* its firing threshold V_{thresh}. This leads to a minimum value condition for $I_{\text{thresh}}^{\text{bias}}$.
- The neuron has to fire a minimum spike rate when V_{rest} is set Δv_{high} *above* V_{thresh}. This leads to another minimum value condition for $I_{\text{thresh}}^{\text{bias}}$.
- The effective reset potential of the neuron is not allowed to differ more than Δv_{reset} from the desired value. This leads to a maximum value condition for $I_{\text{thresh}}^{\text{bias}}$.

All three conditions are tested, i.e. the corresponding voltage parameters are applied and the value of $I_{\text{thresh}}^{\text{bias}}$ is swept while the output firing rate of the analyzed neuron is recorded. The resulting minimum and maximum values are compared, and if there is no valid range left, the neuron is tagged as unusable in terms of this calibration regime. If there is a range of possible values for $I_{\text{thresh}}^{\text{bias}}$, the largest value is chosen for later experiments, because the larger $I_{\text{thresh}}^{\text{bias}}$ is, the smaller are the firing threshold fluctuation due to transistor mismatches.

For the `FHW-1.3-No.17` chip, the tolerance values $\Delta v_{\text{low}} = 5\,\text{mV}$, $\Delta v_{\text{high}} = 5\,\text{mV}$ and $\Delta v_{\text{reset}} = 3\,\text{mV}$ (`BVD`) led to a total of 38 out of the 384 neuron cells which were unusable. On the `FHW-1.3-No.18` chip, for 17 neurons no satisfiable regime could be found, and on the `FHW-1.3-No.25` chip there are 15 unusable cells.

[1]Digital to Analog Converter

The described method is the only way to minimize the firing threshold and reset potential fluctuations, because the voltage parameters themselves cannot be individually configured for every neuron (see Section 2.1.4). A total of four values per parameter and chip are available, which is not sufficient to counterbalance the neuron-to-neuron fluctuations of all 384 cells.

5.2.3 Membrane Time Constant Calibration

For each `FHW-1` neuron circuit, the hardware membrane capacitance C_m cannot be controlled by any programmable parameter (see Section 2.1.4), i.e. it is fixed. The hardware design aims at identical capacitances for all circuits, but due to transistor-level hardware variations (see Section 4.2.1), the real physical capacitances follow a distribution which is not known a priori. For the temporal dynamics of the hardware membrane potentials, the absolute value of these capacitances does not influence the spiking behavior of the neurons, provided that the time constants $\tau_\mathrm{m} = \frac{C_\mathrm{m}}{g_\mathrm{l}}$ (see Section 2.1.2) are equal for every circuit, and that identically configured synapses have the same impact on every membrane. The first condition is addressed in the following.

The leakage conductances g_l can be configured individually for every neuron by tuning the corresponding hardware control current $I_{g_\mathrm{l}}^\mathrm{ctrl}$ (see Section 2.1.4), but it is not possible to directly measure the resulting values of g_l for validation. Even if g_l was perfectly controllable, applying the same value for all neurons would result in fluctuating membrane time constants due to the variability of C_m. Hence, setting the membrane time constants $\tau_\mathrm{m} = \frac{C_\mathrm{m}}{g_\mathrm{l}}$ to desired values is not possible without additional effort.

In order to approximate the desired values for the individual membrane time constants, a feedback loop has been implemented which iteratively determines the actual value of τ_m for every neuron and adjusts $I_{g_\mathrm{l}}^\mathrm{ctrl}$. This is repeated in a *binary search scheme* (Knuth, 1997) until the measured values of τ_m have reached a value within some configurable tolerance range.

A crucial part of this procedure is the method with which τ_m is measured. In Section 4.1.4 a purely spike-based and thus fast measurement method is presented which has been developed specifically for this purpose. Due to the chip malfunctions presented in Section 4.3.2, the parameter programming precision that is necessary for the application of the mentioned method cannot be achieved. Hence, it had to be extended by the following steps:

- For every neuron, a resting and a reset potential $V_\mathrm{rest}^\mathrm{in}$ and $V_\mathrm{reset}^\mathrm{in}$ is written.
- Then, the actually generated values $V_\mathrm{rest}^\mathrm{out}$ and $V_\mathrm{reset}^\mathrm{out}$ are determined with oscilloscope measurements.
- Based upon these two values, the target output value $V_\mathrm{thresh}^\mathrm{out}$ is calculated: $V_\mathrm{thresh}^\mathrm{out} = V_\mathrm{rest}^\mathrm{out} - \frac{1}{e} \cdot (V_\mathrm{rest}^\mathrm{out} - V_\mathrm{reset}^\mathrm{out})$ (see Section 4.1.4).
- With a binary search algorithm and test runs utilizing the oscilloscope, the input value $V_\mathrm{thresh}^\mathrm{in}$ is searched that has to be written in order to actually achieve $V_\mathrm{thresh}^\mathrm{out}$.

With this extra effort invested to tune the firing threshold, the membrane time constant access method described in Section 4.1.4 can be applied to the `FHW-1` system, but it lost its speed advantage due to two reasons:

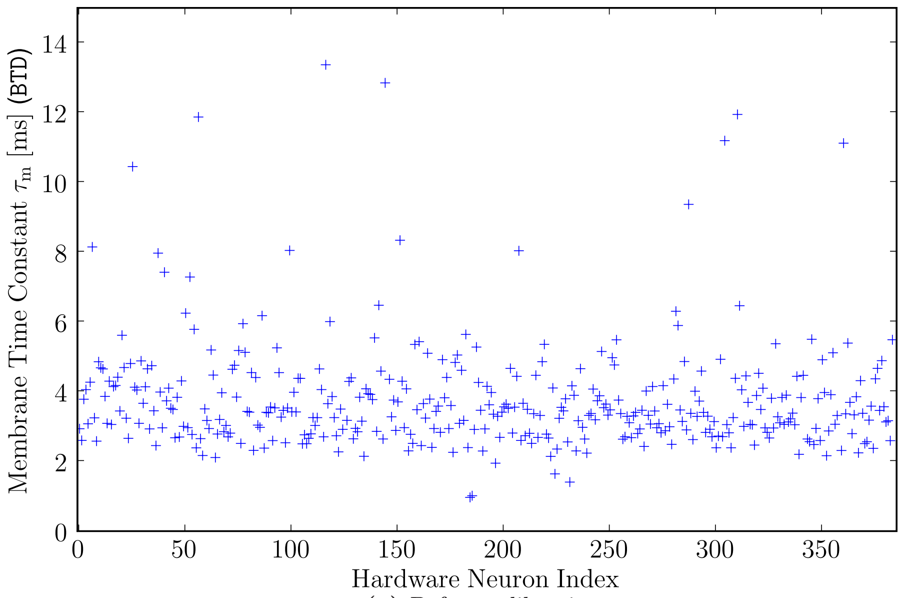
- Due to the necessary utilization of the oscilloscope, the acquisition of the recorded membrane potentials significantly slows down the experiment repetition frequency.

- Since the `FHW-1` system allows access to only a small sub-set of membrane potentials at the same time, a full chip has to be calibrated sub-set by sub-set, while otherwise all neurons could have been calibrated in parallel. The spike recording deadlock issue described in Section 4.3.1 causes the same type of problem.

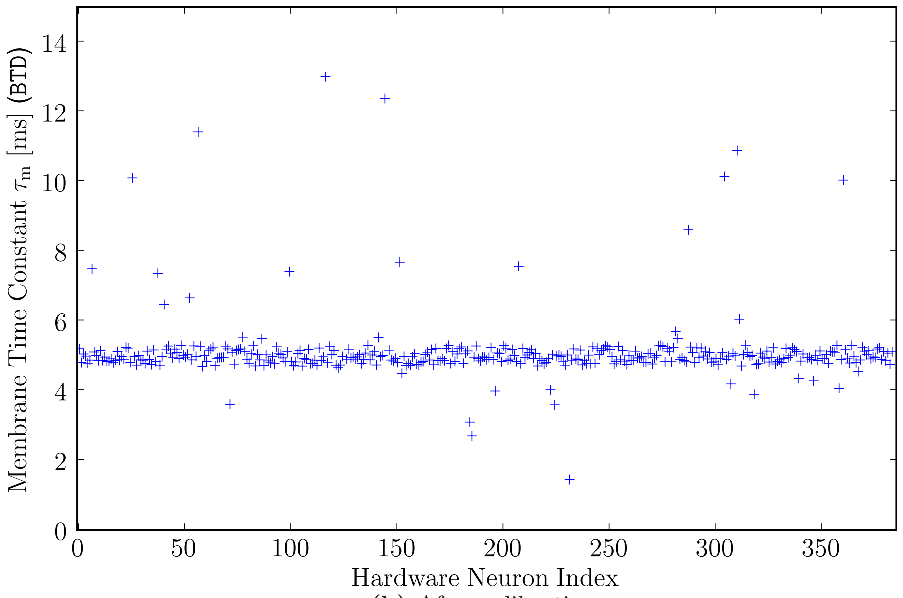
An additional issue that distorts the method described in Section 4.1.4 is the fact that the refractory period τ_{ref} is not known well due to reasons described in Section 4.3.2. Therefore, a constant value of $\tau_{\mathrm{ref}} = 1.0\,\mathrm{ms}$ (`BTD`) has been assumed for all membrane time constant calibrations on the `FHW-1` systems. Nevertheless, deviations in the order of $\pm 0.5\,\mathrm{ms}$ from this generic value have to be expected.

Based upon the described procedure, the membrane time constants of all neurons on all chips currently in operation have been calibrated to target values in the order of $5\,\mathrm{ms}$ (`BTD`). The choice of this value is motivated in Section 4.3.5. Figure 5.2 shows the results of the membrane time constant calibration performed on the `FHW-1.3-No.18` chip by comparing the values of τ_{m} with unmodified, i.e. identical values for g_{l}. Figure 5.3 bins the same data into histograms. Both figures illustrate the significant improvement achieved by the calibration algorithm.

Figure 5.4 shows the values of the leakage conductance control current $I_{g_{\mathrm{l}}}^{\mathrm{ctrl}}$ which are determined by the calibration algorithm on the `FHW-1.3-No.18` system and which result in the improved homogeneity of the membrane time constants depicted in Figure 5.2**(b)**. As an effect of the binary search algorithm (Knuth, 1997) and the chosen logarithmic vertical axis, the resulting control currents form a grid-like structure.

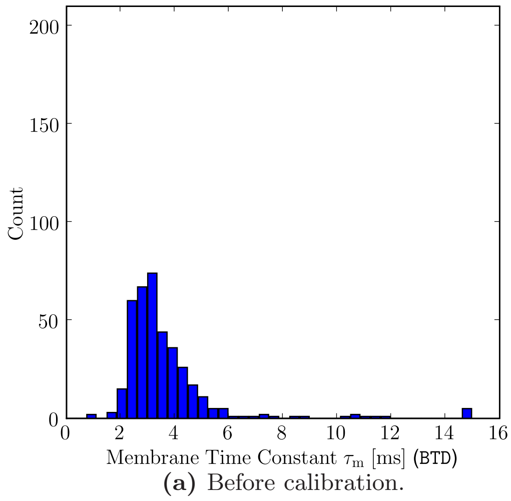


(a) Before calibration.

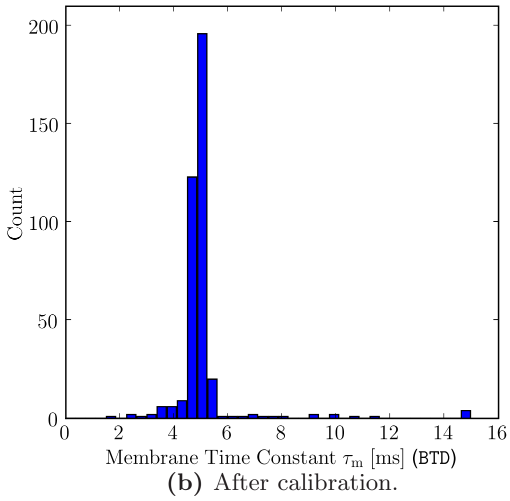


(b) After calibration.

Figure 5.2: Results of the membrane time constant calibration routine described in this section, for all 384 neurons on the `FHW-1.3-No.18` system. The target time constant was $\tau_m = 5.0\,\text{ms}$. A few neurons did not fire at all (see Section 4.3.2 for explanation), for those no data point is plotted. **(a)** Measured membrane time constants without calibration, i.e. with the same value for the leakage conductance parameter applied to all neurons. **(b)** Membrane time constants after calibration.



(a) Before calibration.



(b) After calibration.

Figure 5.3: Histograms of the results shown in Figure 5.2: Membrane time constants as measured on the `FHW-1.3-No.18` chip before **(a)** and after **(b)** calibration. The standard deviation of the plotted values divided by their mean value is $\frac{\sigma}{\mu}(\tau_\mathrm{m}) = 0.42$ for the uncalibrated and $\frac{\sigma}{\mu}(\tau_\mathrm{m}) = 0.19$ for the calibrated neurons.

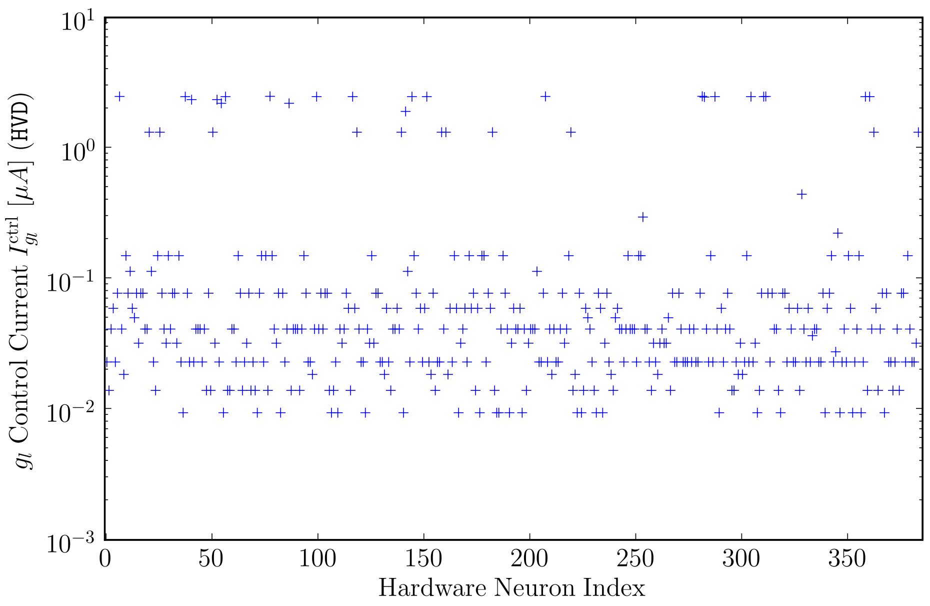


Figure 5.4: The values of the leakage conductance control current $I_{g_l}^{\mathrm{ctrl}}$ as determined by the calibration algorithm described in this section for all 384 neurons on the `FHW-1.3-No.18` system. These heterogeneous values result in the homogeneous membrane time constants shown in Figure 5.2**(b)**. The grid-like structure of the current values is an effect of the binary search algorithm (Knuth, 1997) and the logarithmic scaling of the vertical axis.

5.2.4 Synapse Dynamics Calibration

As elaborately explained in Section 2.1.2, the synaptic signal transmission in the `FHW-1` system is comprised of three stages:

- The output of a pre-synaptic neuron or an external spike source connects to a so-called *synapse driver*. As soon as such a driver receives a digital spike, it generates a linearly first rising and then falling voltage ramp. In an ideal system, the slopes of both ramp sections are configurable, but due to design-related issues (see Section 4.3.3), the slope of the falling edge is not freely configurable in the available versions of the chip.

- Every synapse driver connects to 192 so-called *synapse nodes*, where the generated voltage ramps are transformed into currents. The amplitudes of these exponentially rising and falling current courses depend on the amplitude of the voltage ramp, but also linearly on the weight stored as a four-bit value in every synapse node.

- These currents are routed to post-synaptic neuron circuits, where they control the conductance between the neuron membrane and a reversal potential. Every neuron has two input current lines, one for a conductance towards its excitatory reversal potential, and one for a conductance towards its inhibitory reversal potential. Each line receives and sums up the currents from many synapse nodes.

Sources of Synaptic Variability There are multiple mechanisms involved in this hardware synaptic transmission process that are possibly affected by process-inherent transistor-level variations (see Section 4.2.1). The main candidates which are believed to result in deviations from ideal synaptic dynamics are:

- A driver-to-driver variability in the effect of the control currents $I_{\tau_{\mathrm{rise}}}^{\mathrm{ctrl}}$ and $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ for the rising and falling voltage ramps generated by the synapse drivers (see Section 2.1.4 for an explanation of the hardware control parameters).

- A driver-to-driver variability in the effect of the control current $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ which determines the amplitude of this voltage ramp.

- A limited sensitivity of the comparator which detects the maximum amplitude of the voltage ramp and triggers the change from rising to falling slope.

- A node-to-node variability in the transformation of the voltages into currents in the synapse nodes.

- A neuron-to-neuron variability in the effect of these currents onto the conductance between the neuron membrane and the reversal potential.

- A neuron-to-neuron variability in parasitic capacitances inherent the lines which deliver these currents.

This list considers only possible distortions of static dynamics, i.e. of synapses which are not configured to exhibit depression of facilitation (see Section 2.1.3, “Synaptic Depression and Facilitation”). For synapses with such mechanisms enabled, more hardware-specific factors can impose additional fluctuations.

Counterbalancing Principles

In the following, paradigms are listed which form the basis upon which the synapse driver calibration algorithm is implemented.

Tuning Granularity Every synapse driver receives its own parameter values for the control currents $I_{\tau_{\rm rise}}^{\rm ctrl}$, $I_{\tau_{\rm fall}}^{\rm ctrl}$ and $I_{\rm amp}^{\rm ctrl}$. Hence, in contrast to the massively shared parameter values for e.g. the firing threshold voltages, these parameters can be utilized to counterbalance the effect of transistor-level variations occurring from synapse driver to synapse driver. Consequently, the first three issues in the above list can be approached with the manipulation of the synapse driver parameters.

For the last two issues in the list, no calibration routine can be applied, because no parameters are available in the `FHW-1` system which are individually configurable for every neuron or every synapse node and which are not fully determined by other constraints. The only exception to this statement are the synaptic four-bit weights, and indeed they are used to homogenize the neuronal responsiveness (see Section 5.2.5), although this reduces the remaining ranges of free synaptic programmability. In this section, though, a calibration algorithm is introduced which approaches only the synapse drivers.

Optimization Objective Besides the output spike times of all neurons in the network and the synaptic weights (interesting e.g. in the context of STDP, see Section 1.3.3), the only activity information accessible from a `FHW-1` device during operation is the membrane potential of selectable neurons (see Section 2.1.5). This implies that a direct measurement of the conductance courses at the synapse drivers, which would be the optimal information for a corresponding calibration, is not possible. Hence, the following objective for a homogenization of synapse driver efficacies is proposed: The effect that a single spike running into a synapse has on the post-synaptic membrane potential (the so-called *post-synaptic potential* or PSP) is supposed to be equal within a provided tolerance range.

Since the rise time of the synaptic conductance courses is kept fixed (see Section 2.1.4 for an explanation), the amplitude and the fall time remain as free parameters. Due to the `FHW-1` design-inherent malfunction described in Section 4.3.3, the control currents for the synaptic fall times are restricted to values below $0.15\,\mu$A (`HVD`) in the available first three versions of the chip. Still, the tuning of this current is integrated into the implemented calibration routine for future versions of the hardware. Consequently, since the temporal aspects of the synaptic conductance courses can only be controlled inadequately, the amplitude of these courses itself is not an optimal optimization criterion for the synaptic efficacy. Therefore, the chosen major objective of the synapse driver calibration is to homogenize the integral over the PSPs generated by all different synapse drivers connected to the same neuron membrane.

Utilizing NEST as a Reference Based on the PyNN approach (see Section 3.1.3), the synapse driver calibration paradigm is even extended: The calibration setup and execution is done with PyNN, and based upon this description, a reference run with the software simulator NEST (see Section 3.1.4) provides the target value for all hardware tuning.

Basis Activity Regime For a number of reasons, the PSP acquisitions of a synapse driver calibration have to be performed on a neuron membrane under significant stimulation:

- As outlined above, the full path of synaptic signal transmission in hardware incorporates a variety of electronic circuitry, wires and switches. The resulting parasitic capacitances have a dumping effect, especially when they have to be initially charged after a period without signal transmission. This effect can be minimized by generating a basis activity via the electric signal path, i.e. by sending spikes into the neuron via the synapse driver that is to be calibrated. In such an active regime, the observed PSPs are subject to less distortions.

- The target PSP amplitudes are typically in the same order or below the level of noise on an `FHW-1` membrane potential. Therefore, the technique of spike-triggered averaging (STA) as introduced in Section 4.1.1 is applied in order to access the PSP. This requires a high number of membrane potential samples triggered by input spikes, and in order to establish a membrane potential with reasonable fluctuations and a controllable average value (which influences the averaged PSP shape, as can be seen e.g. from Equation 2.1), both excitatory and inhibitory spike sources with high event rates have to be applied.

- If the time constants of the acquired PSPs shall be interpreted as the time constants of the underlying conductance course (which is an interesting additional information, but not crucial for a well working calibration routine), the observed membrane must be in a high-conductance state (see Section 1.3.1). This requires a high level of synaptic stimulation. In Section 4.1.2, a corresponding test method is introduced.

- Due to the design-related malfunction described in Section 4.3.4 (the excitatory reversal potential significantly drops under heavy load), a load for excitatory network activity has to be applied that is close to the load expected in the experiment for which the chip is calibrated. Since the activity dependence is strong, this issue actually makes a reliable calibration impossible, because every synapse driver setup is only valid for a very narrow range around the conditions under which it was calibrated. This issue is considered to be not solvable without re-designing the chip.

The Calibration Algorithm

The setup utilized by the implemented synapse driver calibration algorithm is the following: A single neuron is exposed to $N_\mathrm{e}^\mathrm{stim}$ externally generated Poisson-type spike trains via excitatory synapses and to $N_\mathrm{i}^\mathrm{stim}$ Poisson-type spike trains via inhibitory synapses. These spike trains are statistically independent, and each of them fires with an average rate of f_stim over a period T_exp. The weights of the excitatory (inhibitory) synapses, i.e. their peak conductances, are all set to the same value g_e (g_i). The firing mechanism of the stimulated neuron is deactivated, its membrane potential is recorded.

Step 1: Establish Working Point in NEST In a first phase, a selectable average membrane potential is established, with reasonable membrane potential fluctuations over time. Therefore, the setup, which is described in PyNN, is repeatedly executed in the software simulator NEST. If not directly involved in the calibration routine, all neuron and synapse parameters are set to values as they are expected to be applied during future experiments. Then, the weights of the inhibitory spike trains g_i are set to zero, and the weights of the excitatory spike trains g_e are increased from iteration to iteration, starting at zero. In every run, the resulting average membrane potential $\langle V(t) \rangle$ is determined. The increasing of g_e is stopped as soon as $\langle V(t)\rangle$ reaches the firing threshold voltage of the neuron. (As mentioned above, the firing mechanism itself is deactivated.) The corresponding value of the excitatory weight $g_{\mathrm{e}}^{\mathrm{wp}}$ is kept, then g_{i} is iteratively increased. Again $\langle V(t)\rangle$ is checked after every iteration, and as soon as the selectable working point, typically $\langle V(t)\rangle^{\mathrm{wp}} = V_{\mathrm{rest}} + 2/3 \cdot (V_{\mathrm{thresh}} - V_{\mathrm{rest}})$, is reached, the corresponding inhibitory weight $g_{\mathrm{i}}^{\mathrm{wp}}$ is also kept.

This procedure provides a well defined working point of the membrane potential which exhibits the desired average value, but also well defined fluctuations caused by a balanced influence of both excitatory and inhibitory spike trains. The average PSP caused by one synapse on a membrane with this basis activity represents the target shape the hardware shall be able to approximate with every synapse driver.

Step 2: Determine PSP Integrals and Decay Time Constants in NEST With these synaptic weights $g_{\mathrm{e}}^{\mathrm{wp}}$ and $g_{\mathrm{i}}^{\mathrm{wp}}$ for the external stimulation, multiple runs of the basic setup are performed, with new randomly generated spike trains for each repetition. The spike-triggered averaging technique is applied to extract the average PSP generated by every synapse in this regime. For both the excitatory and the inhibitory average PSP, the integral $I_{\mathrm{PSP}} \equiv \int (V_{\mathrm{PSP}}(t) - \langle V(t)\rangle \; \mathrm{d}t$ and, via an exponential fit, the decay time constant τ_{PSP} are determined (see Figure 5.5). The PSP integral and decay time constant acquired with NEST are the target values for the following hardware calibration.

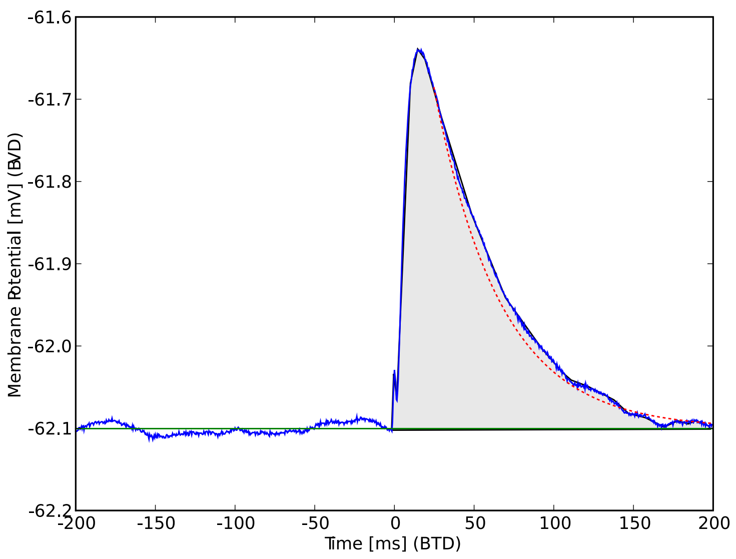


Figure 5.5: For every synapse driver, the calibration algorithm determines the integral and the decay time constant of an average PSP which has been acquired with the STA technique. The corresponding working point voltage $\langle V(t)\rangle$ is estimated by the average value of the first half of the PSP trace (horizontal line). The sum over all trace values in the right half minus the base line (grey area) delivers the value of the PSP integral I_{PSP}. An exponential fit from 10 ms (`BTD`) after the maximum of the PSP curve provides the decay time constant estimator τ_{PSP} (dashed line).

Step 3: Establish Working Point in Hardware Before the first synapse driver calibration has been performed, no rule is available which allows to translate biological synapse parameters into the corresponding hardware control parameters. This is achieved by applying exactly the same procedure as in Step 1, but instead of tuning g_{e} (g_{i}), the hardware control currents $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ for all $N_{\mathrm{e}}^{\mathrm{stim}}$ excitatory ($N_{\mathrm{i}}^{\mathrm{stim}}$ inhibitory) synapse drivers are collectively swept. When the biological interpretation of the hardware average membrane potential (see Section 3.1.5) has reached the working point $\langle V(t)\rangle^{\mathrm{wp}}$, the values of $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ are found which correspond to $g_{\mathrm{e}}^{\mathrm{wp}}$ respectively $g_{\mathrm{i}}^{\mathrm{wp}}$. The corresponding *basis translation factors* $T_{\mathrm{e}}^{\text{bio-hw}}$ and $T_{\mathrm{i}}^{\text{bio-hw}}$ are stored and kept for all future experiments. Still, such a translation can only establish the desired working point, but does not yet provide a homogenization of the synapse drivers efficacies. This is done in the next step, where additional calibration factors are searched for every individual synapse driver.

Step 4: Determine and Tune PSP Integrals in Hardware Like in Step 3, the integrals and decay time constants are determined for the average PSP of every synapse driver acquired with the STA technique. Now, this is done in an iterative loop, and the values of both $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ and $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ are adjusted after every repetition in order to optimize the matching between the measured values and the target values determined in Step 2.

This individual fine-tuning of the synapse driver parameters possibly shifts the membrane working point. Therefore, after every repetition, an adjustment control procedure re-establishes the working point by collectively increasing or decreasing the individual calibration factors by the same factor.

Results and Discussion

After a maximum number of iterations or after all synapse drivers have reached their calibration targets, for every driver s two individual calibration factors are available: $c_{\mathrm{eff},s}$ and $c_{\tau,s}$. The parameter value of $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ for every driver is from then on calculated by

$$\begin{aligned} I_{\mathrm{amp},s}^{\mathrm{ctrl}} &= g_{\mathrm{e}}^{\mathrm{wp}} \cdot T_{\mathrm{e}}^{\text{bio-hw}} \cdot c_{\mathrm{eff},s} \quad \text{for excitatory drivers} \quad , \\ I_{\mathrm{amp},s}^{\mathrm{ctrl}} &= g_{\mathrm{i}}^{\mathrm{wp}} \cdot T_{\mathrm{i}}^{\text{bio-hw}} \cdot c_{\mathrm{eff},s} \quad \text{for inhibitory drivers} \quad . \end{aligned} \tag{5.2}$$

Note that this formula is only valid for the specific values of $g_{\mathrm{e}}^{\mathrm{wp}}$ and $g_{\mathrm{i}}^{\mathrm{wp}}$ at which the calibration has been performed. Due to the unwanted activity dependence of the synaptic efficacy (see Section 4.3.4), no relation between $g_{\mathrm{e}}^{\mathrm{wp}}$ and $I_{\mathrm{amp},s}^{\mathrm{ctrl}}$ can be given that is valid for a whole set of different scenarios. As long as the underlying issue is not solved, the presented calibration has to be repeated for every new type of experiment.

For the synaptic time constants, the calculation of the individual parameter values of $I_{\tau_{\mathrm{fall}},s}^{\mathrm{ctrl}}$ is analogous to Equation 5.2, i.e. the values are determined by a basis translation factor and by individual calibration factors. Since the adjustable range of $I_{\tau_{\mathrm{fall}},s}^{\mathrm{ctrl}}$ is extremely limited due to the hardware problem described in Section 4.3.3, the results achieved on the `FHW-1.3` systems hardly improve the decay time constant homogeneity and thus are not shown here.

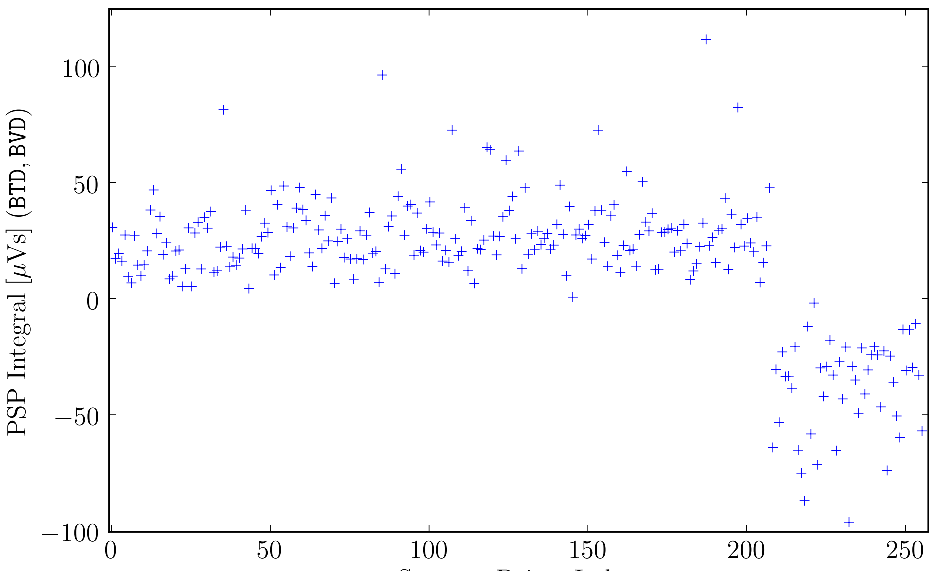
Figure 5.6 shows the results of such a calibration routine applied to the left network block of the `FHW-1.3-No.18` chip. The numbers of excitatory and inhibitory Poisson-type spike trains were chosen to be $N_{\mathrm{e}}^{\mathrm{stim}} = 208$ and $N_{\mathrm{i}}^{\mathrm{stim}} = 48$. This specific partitioning was motivated by the kind of experiments planned with the calibrated system, which will be presented in Section 6.2.1. In sub-figure **(a)**, the hardware PSP integrals of all 256 synapse drivers after

Step 3 are plotted. The excitatory and inhibitory drivers can clearly be recognized by the sign of the integral. In sub-figure **(b)**, the PSP integrals of the same synapse drivers, but after Step 4 are shown. The improvement of the integral homogeneity is obvious, and the matching of most of the values with the target range (horizontal lines) can be observed.

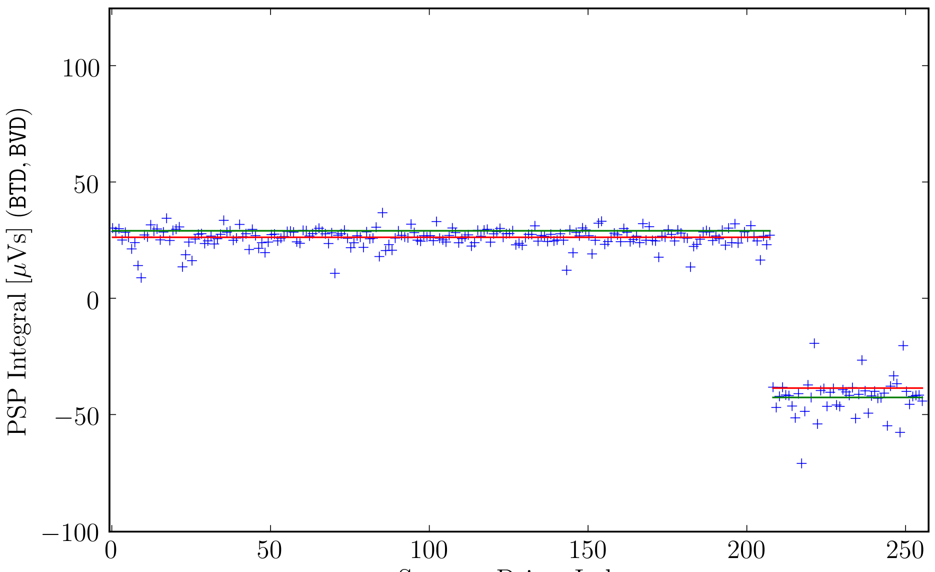
In Figure 5.7, the PSP integrals of the excitatory synapse drivers shown in Figure 5.6 are binned into histograms, one for the situation after Step 3 **(a)** and one after Step 4. It is obvious that the width of the distribution is strongly narrowed by the calibration algorithm, the distribution width changes from $\frac{\sigma}{\mu}(I_{\mathrm{PSP}}) = 0.56$ before the calibration to $\frac{\sigma}{\mu}(I_{\mathrm{PSP}}) = 0.15$ after the calibration.

Due to the design-related malfunction of the `FHW-1` systems described in Section 4.3.4, a global drift of the $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ parameters towards large values worsens the load on the excitatory reversal potential. For the described calibration scenario, this can even result in an inverted dependency of the PSP integral on the $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ parameter, i.e. from a certain value, increasing the parameter reduces the integral. This issue is solved by applying a maximum value $I_{\mathrm{amp}}^{\mathrm{max}}$ for $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ which is much smaller than the maximum that is technically possible (see Section 2.1.4). This technique massively improves the final result of the calibration algorithm, but for a few synapse drivers it means that the target value cannot be achieved.

Figure 5.8 shows how the values of $I_{\mathrm{amp}}^{\mathrm{ctrl}}$ are distributed after Step 4. It can be observed that the values are distributed across the full width of the available range ($0.002\,\mu\mathrm{A} \leq I_{\mathrm{amp}}^{\mathrm{ctrl}} \leq I_{\mathrm{amp}}^{\mathrm{max}} = 0.14\,\mu\mathrm{A}$), and that for some synapse drivers the upper bound was probably a real limit during the calibration process.



(a) Before calibration.



(b) After calibration.

Figure 5.6: Results of the synapse driver efficacy calibration routine described in this section, for all 256 synapse drivers on the left network block of the `FHW-1.3-No.18` system. The target integral of an average PSP was $I_{\mathrm{PSP}}^{\mathrm{exc}} = 27.7\mu\,\mathrm{Vs}$ for the excitatory synapse drivers (indices 0 – 207) and $I_{\mathrm{PSP}}^{\mathrm{inh}} = -39.9\mu\,\mathrm{Vs}$ for the inhibitory synapse drivers (indices 208 – 255). **(a)** Measured PSP integral without calibration, i.e. with the same value for the synaptic conductance course amplitude parameter applied to all drivers. **(b)** PSP integral after calibration. The two vertical lines denote the upper and the lower tolerance limits which were accepted by the calibration algorithm. For those values still outside of this range, no better calibration value could be found.

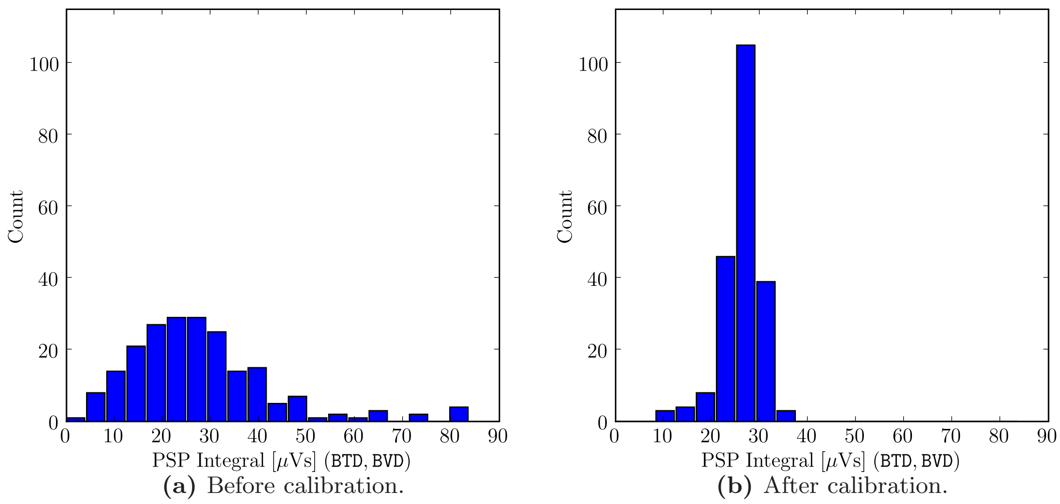


Figure 5.7: Histograms of the results shown in Figure 5.6 (excitatory only): PSP integrals I_{PSP} caused by the excitatory synapse drivers on the left network block of the `FHW-1.3-No.18` chip before **(a)** and after **(b)** calibration. The standard deviation of the plotted values divided by their mean value is $\frac{\sigma}{\mu}(I_{\text{PSP}}) = 0.56$ for the uncalibrated and $\frac{\sigma}{\mu}(I_{\text{PSP}}) = 0.15$ for the calibrated case.

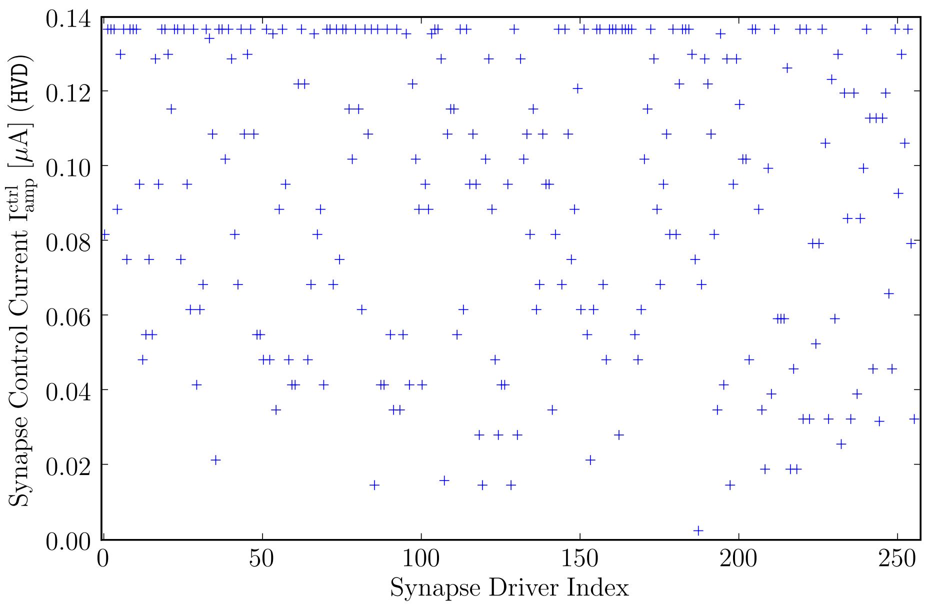


Figure 5.8: The values of the synapse driver control current $I_{\text{amp}}^{\text{ctrl}}$ as determined by the calibration algorithm described in this section for all 256 synapse drivers on the left block of the `FHW-1.3-No.18` system. These heterogeneous values result in the homogeneous PSP integrals shown in Figure 5.7**(b)**.

5.2.5 Synapse Weights Calibration

As mentioned in Section 5.2.3, the capacitances of the hardware neuron circuits are subject to transistor-level variations. This cannot be directly counterbalanced by any calibration mechanism, since there is no parameter which allows to manipulate these capacitances. Instead, as described in Section 5.2.3, the *membrane time constant* is homogenized via the individually adjustable leakage conductance g_l. But this calibration method only minimizes the impact of the capacitance variations on the leakage mechanism, as can be seen in the first term on the right hand side of Equation 2.1. The remaining two terms, which model the synaptic influence on the temporal evolution of the membrane potential, are still affected by the neuron-to-neuron variability of membrane capacitances. Furthermore, they are affected by neuron-to-neuron fluctuations of the conductances between membranes and the reversal potentials.

The only hardware parameter which allows to tune the synaptic efficacies individually for every neuron are the synaptic weights stored as four-bit values in every synaptic node (see Section 2.1.2). Although these parameters do not seem to be qualified for any kind of fine-tuning due to their coarse value resolution, the fact that every neuron typically receives synaptic input via many of these nodes allows a statistical approach. As already indicated in the description of the weight discretization process in Section 3.1.5, a large number of synapses per neuron provides a fine adjustability of the average synaptic input. Hence, at least by means of this average value, a counter-balancing method for neuron-to-neuron variabilities of synaptic influence has been developed.

The Calibration Algorithm

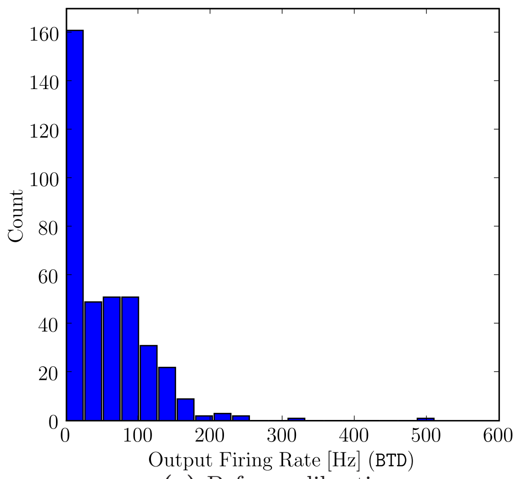
The following calibration method has been developed and implemented: For every neuron n on an `FHW-1` chip, a so-called *synapse weight factor* $c_{\omega,n}$ is created, with a default value of 1. This factor is multiplied to the weight of every *excitatory* synapse, i.e. it directly influences the total amount of excitatory stimulation to neuron n. A basic experimental setup is described in PyNN: A neuron receives a set of Poisson-type input spike trains with well defined firing rates and synaptic weights, some of which are excitatory, some inhibitory. A set of reference runs in NEST (with freshly generated spike trains per repetition) reveals a reliable target output rate of the neuron. Then, for every neuron on the `FHW-1` chip to be calibrated, the same input patterns like for the NEST reference runs are applied, utilizing the same calibrated synapse drivers for every neuron (see Section 5.2.4).

In an iterative loop, the output rate of every neuron is measured and compared with the target rate. The value of $c_{\omega,n}$ is adjusted adequately in every iteration, until either all output rates have reached a tolerance range around the target value, or until a maximum number of iterations is exceeded.

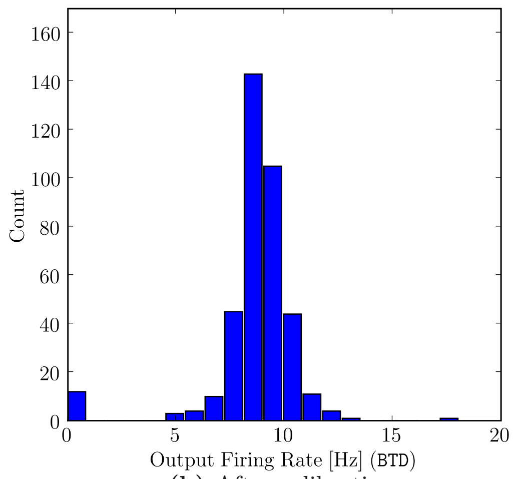
Results and Discussion

Figure 5.9 shows the result of such a calibration routine applied to the `FHW-1.3-No.18` chip. In sub-figure **(a)**, a histogram of all measured output firing rates with a synapse weight factor of $c_{\omega,n} = 1$ for all neurons is shown. Sub-figure **(b)** shows the distribution of the firing rates after the values of $c_{\omega,n}$ have been individually optimized in order to achieve the NEST reference output rate of $f_{\text{out}}^{\text{NEST}} = 8.96\,\text{Hz}$ (`BTD`). Note the different scaling of the frequency axes! Before the calibration has been applied, the mean and standard deviation over all neurons on the chip was $(\mu \pm \sigma)(f_{\mathrm{out}}^{\mathrm{chip}}) = (55 \pm 59)\,\mathrm{Hz}$ (BTD). After the optimization, it is $(\mu \pm \sigma)(f_{\mathrm{out}}^{\mathrm{chip}}) = (8.7 \pm 2.0)\,\mathrm{Hz}$ (BTD).

This result has to be interpreted carefully. As described in Section 4.3, multiple hardware neurons do emit spikes which do not have a correspondence in the biological reference model. This can be e.g. caused by the so-called *multi-spike* issue, where multiple hardware events are emitted for one single firing threshold crossing (see Section 4.3.6). Unwanted spikes can also result from a highly distorted firing threshold due to transistor-level variations (see Section 4.3.2).



(a) Before calibration.



(b) After calibration.

Figure 5.9: Histograms of the average output firing rates over all neurons of the FHW-1.3-No.18 chip before **(a)** and after **(b)** calibration of the *synapse weight factors*. The target output rate provided by a NEST reference simulation was $f_{\mathrm{out}}^{\mathrm{NEST}} = 8.96\,\mathrm{Hz}$. Note the different horizontal scalings. The mean and the standard deviation over all plotted values is $(\mu \pm \sigma)(f_{\mathrm{out}}^{\mathrm{chip}}) = (55 \pm 59)\,\mathrm{Hz}$ (BTD) for the uncalibrated and $(\mu \pm \sigma)(f_{\mathrm{out}}^{\mathrm{chip}}) = (8.7 \pm 2.0)\,\mathrm{Hz}$ (BTD) for the calibrated case.

Figure 5.10 shows a histogram of the *synapse weight factors* determined by the calibration routine. As expected from sub-plot **(a)** in Figure 5.9, where the average output firing rate before calibration is shown to be significantly larger than the target value, the average value of the tuned factors $c_{\omega,\mathrm{n}}$ is smaller than 1.

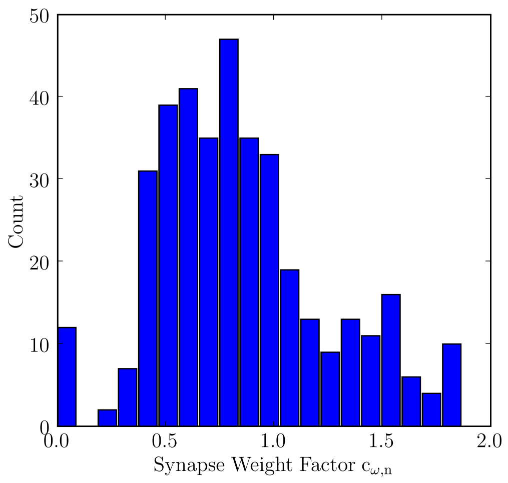


Figure 5.10: Histogram of all *synapse weight factors* $c_{\omega,\mathrm{n}}$ as determined by the output firing rate homogenization for the `FHW-1.3-No.18` chip (see Figure 5.9). The mean and standard deviation over all values are $(\mu \pm \sigma)(c_{\omega,\mathrm{n}}) = 0.85 \pm 0.40$.

5.2.6 Calibration Reproducibility and Portability

Production process-inherent transistor-level variations are assumed to cause fluctuations in the characteristics of circuit dynamics. They are believed to be the main reason for the necessity of calibration methods like those presented in the previous sections. Still, there are other possible effects which can result in circuit inhomogeneities, e.g. different surrounding areas of identical circuitry, which possibly impose different parasitic capacitances or which hold different sources of crosstalk.

The transistor-level effects are randomly distributed, i.e. if the corresponding distortions on two chips of the same design are compared, no repeating pattern can be expected to be found. Systematic parasitic effects, though, are design-related and might be found in every individual chip with very similar characteristics and spatial patterns. If such systemic effects were strongly dominating the transistor-level distortions, calibration data acquired on one chip could be used for all other chips as well. The following calibration data comparisons address this question and show that both the dominance of design-related distortions and the dominance of process-inherent random effects are possible.

Another issue of hardware imperfection is the possible temporal development of factors which can influence the hardware behavior, such as temperature or electro-magnetic fields from close-by devices. Therefore, the following two examples also refer to the question of permanent validity respectively reproducibility of calibration data.

Chip-To-Chip Comparison of Membrane Time Constant Calibration Data

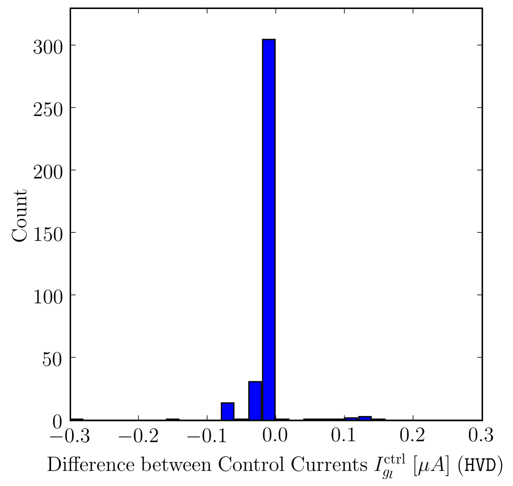
The calibration mechanism presented in Section 5.2.3 determines individual leakage conductance control currents $I_{g_l,n}^{\mathrm{ctrl}}$ for every neuron n. These values have been determined, among others, for the `FHW-1.3-No.17` and two times for the `FHW-1.3-No.18` system. In the latter case, the two calibration runs lie months apart.

For the resulting three sets of calibration data, the values of the tuned parameter $I_{g_l}^{\mathrm{ctrl}}$ are compared for every neuron. Figure 5.11 bins the differences between the individual values $\Delta I_{g_l,n}^{\mathrm{ctrl}}(\mathrm{chip1,chip2}) \equiv I_{g_l,n}^{\mathrm{ctrl}}(\mathrm{chip1}) - I_{g_l,n}^{\mathrm{ctrl}}(\mathrm{chip2})$ into histograms. It compares

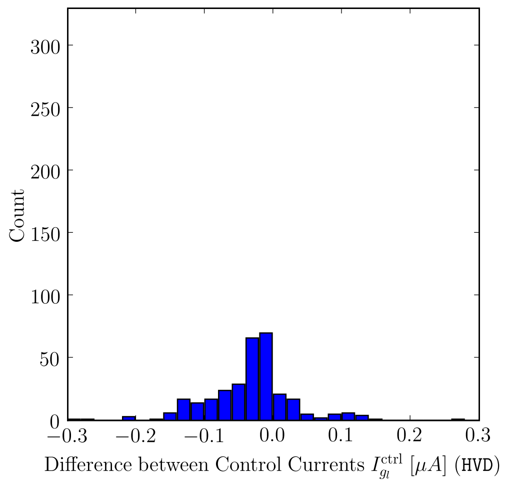
- the first and the second calibration run on the `FHW-1.3-No.18` system in sub-figure **(a)**,
- one calibration run on the `FHW-1.3-No.18` system and one calibration run on the `FHW-1.3-No.17` system in sub-figure **(b)**.

It is obvious that for two calibration runs on the same chip, the fluctuation of the resulting calibration values is much smaller than for calibration runs on two different chips. The standard deviation of the plotted current differences is $\sigma(\Delta I_{g_l,n}^{\mathrm{ctrl}} = 0.030\,\mu\mathrm{A}$ (`HVD`) for the same-chip comparison, and $\sigma(\Delta I_{g_l,n}^{\mathrm{ctrl}} = 0.065\,\mu\mathrm{A}$ (`HVD`) for the different-chips comparison. The fluctuation of the calibration values themselves is $\sigma(I_{g_l,n}^{\mathrm{ctrl}}) = 0.5\,\mu\mathrm{A}$ (`HVD`) for the `FHW-1.3-No.18` system.

Conclusions For the considered devices and for the membrane time constant calibration, a significant fraction of the counterbalanced distortions is independent of the device. This can be deduced from the differences between the calibration data sets for two different chips compared to the fluctuations of each individual set. Obviously, the location of a neuron circuit on the chip has a large impact on its membrane time constant, i.e. the transistor-level variations are dominated by systematic parasitic effects.



(a) `FHW-1.3-No.18` – `FHW-1.3-No.18`



(b) `FHW-1.3-No.18` – `FHW-1.3-No.17`

Figure 5.11: Histograms of the differences between the tuned control parameters $I_{g_l}^{\mathrm{ctrl}}$ for all neurons on an `FHW-1` device acquired from two calibration runs on the same chip **(a)**, between the tuned parameters values acquired from two different chips **(b)**.

Although this suggests the portability of calibration data, an individual calibration per chip does improve the result further, as can be seen from the even smaller differences between two calibration data sets acquired on the same chip. Therefore, a chip-individual calibration is recommended. If any temporal effect is the reason for the remaining differences between two calibrations on the same device, it can most probably be ignored, since the differences in the tuning parameter are close to the resolution limit with which it can be written.

Chip-To-Chip Comparison of Voltage Generator Calibration Data

As stated in Section 5.2.1, the assumed reason for the fluctuations of the input-output relations of the `FHW-1` voltage generators is the process-inherent imprecision of polysilicon resistor production. Therefore, in contrast to the fluctuations of membrane time constants (see section above), which are clearly dominated by an effect related to the location of the circuits on the chip, the voltage generator calibration data is expected to be not transferable from one chip to another at all.

The following observation supports this assumption: The standard deviation of the input-output translation parameter m_{U} (see Section 5.2.1) over all calibrated voltage generators is $\sigma(m_{\mathrm{U}}) = 6 \cdot 10^{-3}$ for the `FHW-1.3-No.17` and for the `FHW-1.3-No.18` system, while the fluctuation of the differences between the corresponding values of m_{U} between these two chips is $\sigma(\Delta m_{\mathrm{U}}) = 7 \cdot 10^{-3}$.

Conclusion The considered voltage generator calibration data is very device-specific. The dominating effect that causes the fluctuations is not related to the location of the circuit on the chip, but to process-inherent random imperfections of the utilized micro-electronic components.

5.3 Measures for Cross-Platform Evaluation

Establishing neuromorphic hardware as a useful component within the neural network modelers' toolbox requires the proof that the hardware system can be operated in biologically realistic regimes. Furthermore, its operability by non-hardware-experts has to be ensured. The interest of the computational neuroscience community for neuromorphic hardware systems will significantly grow if it can be shown that, for a relevant type of experiment, a hardware device can generate the same results like an established software simulator. This has to be shown in terms of measures which satisfy the modelers' needs, while the simulator is outperformed in at least one aspect that is crucial for the chosen type of experiment. Candidates for such aspects which are often mentioned in the context of neuromorphic hardware devices are speed at large model sizes, low power consumption and the possibility to embed them into autonomous systems like robots (Lewis et al., 2000; Netter and Franceschini, 2002; Delbrück and Liu, 2004; Schemmel et al., 2007; Fu et al., 2008).

For the type of neuromorphic devices utilized throughout this thesis, the non-expert usability condition has been met by integrating the hardware interface into the simulator-independent modeling language PyNN (see Davison et al., 2008 and Section 3.1.3). This approach does not only introduce the hardware into the world of established software simulators by means of experiment portability, but utilizing platform-independent descriptions also provides a way to approach the proof of its biological relevance: Based on PyNN, the *same* evaluation measures for relevant experiments can be applied within this unified framework. There can be *one* analysis tool chain on top of *one* model description, finally exposing the hardware system to the necessary beneficial criticism of a community not limited to microelectronics engineers.

In the following, two types of methods are introduced which provide ways to compare the output of spiking hardware models like the `FHW-1` and the `FHW-2` system with the output of software simulators. First, a measure for the *comparison of single neuron spiking activity* is described. It can be utilized to e.g. quantify the output spike timing precision of a hardware neuron from run to run (see Section 6.1.2), or, more generally, to determine the intrinsic noise of a neuron. Based upon that, a software model can be tuned towards the hardware behavior or vice versa. The second type of measures presented below are *statistical descriptors*. They are intended to be applied to recurrent networks of neurons, where they are usually much more useful for an interpretation of the observed dynamics than the raw set of all precise spike times. Statistical treatment can be beneficial for nearly any kind of spike data, but it is especially useful for the output of hardware systems which, due to reasons listed in Sections 4.2 and 4.3, can never be expected to generate exactly reproducible data. In recurrent networks, slight changes in an early state can cause very different activity trajectories within the system's state space, while appropriate measures will remain the same in such a situation. Statistical measures are a necessary step of abstraction to cope with hardware imperfections. They will be applied e.g. in experiments presented in Section 6.2.1.

5.3.1 Spike Train Comparison

Comparing two spike trains is a task which does not only arise when comparing a hardware neuron with a software simulation, but also when trying to determine the intrinsic noise of a real or a hardware neuron by recording one cell and repeating the same stimulation protocol, or when comparing different numerical simulations, to mention just a few situations.

The most basic approach of assessing the the similarity between two spike trains is to compare the number of fired spikes. This method completely neglects all aspects of temporal structure like the synchrony of firing, which, for example in a coincidence detection task, might be much more important than the pure spike count. Therefore, this simple approach is skipped. A typical way to introduce a basic synchrony test is to provide an array of time bins into which the spikes are sorted, and then count the number of identically filled bins. But binning imposes possible discretization artefacts, especially when it comes to systematic time shifts between the spike trains to be compared. Hence, such a technique is not utilized in this context, either.

Based upon work presented in (Victor and Purpura, 1996), the authors of (van Rossum, 2001) introduced a measure for the distance between two spike trains which is well applicable for the comparison between hardware and software simulation results:

Given two original spike trains

$$f_i(t) = \sum_{s=1}^{N_i} \delta(t - t_s)\,, \quad i \in \{1,2\} \qquad , \tag{5.3}$$

the delta peaks are replaced by exponential decays as follows:

$$f_i^{\text{tail}}(t) = \sum_{s=1}^{N_i} H(t - t_s)\, e^{-(t-t_s)/\tau_c}\,, \quad i \in \{1,2\} \qquad . \tag{5.4}$$

$H(t)$ is the Heaviside step function, and τ_c is a free parameter. The distance between $f_1(t)$ and $f_2(t)$ is then defined as

$$D^2(f_1, f_2, \tau_c) = \frac{1}{\tau_c} \int_0^\infty \left[f_1^{\text{tail}}(t) - f_2^{\text{tail}}(t)\right]^2 \mathrm{d}t \qquad . \tag{5.5}$$

E.g. in Section 6.1.3, this measure is used to classify the similarity of a set of hardware runs with one reference software simulation.

5.3.2 Statistical Descriptors of Network Activity

In Section 6.2.1, experiments based on cortical network models inspired by the work presented in (Brunel, 2000) and (Kumar et al., 2008) will be described. They will be analyzed by statistical measures which are described in the following.

In (Kumar et al., 2008), different states of network dynamics are discriminated by:

1. The mean population firing rate.
   It is estimated by the total spike count S_{tot} over the whole population divided by the simulation time T_{sim} and the number N_{net} of neurons:

$$\nu_{\text{net}} = \frac{S_{\text{tot}}}{T_{\text{sim}}\, N_{\text{net}}} \qquad . \tag{5.6}$$

2. The firing rate diversity within the network.
   The individual firing rate of a neuron i is estimated by the total spike count S_i fired by this neuron divided by the simulation time T_{sim}:

$$\nu_{\text{i}} = \frac{S_{\text{i}}}{T_{\text{sim}}} \qquad . \tag{5.7}$$

The standard deviation over these individual firing rates

$$\sigma\left(\nu_i\right) = \sqrt{\frac{\sum_{i=1}^{N_{\text{net}}}\left(\nu_i - \overline{\nu}\right)}{N_{\text{net}}}} \tag{5.8}$$

is a measure for the activity diversity in the network.

3. The synchrony among pairs of neurons.
Given a bin width Δ_{bin}, binning the output spike times of neuron i results in an array $C_i(\Delta_{\text{bin}})$ of spike counts. An estimator for the synchrony between neuron i and j is the correlation coefficient of their joint spike counts C_i and C_j,

$$\text{Corr}(C_i, C_j) = \frac{\text{Cov}(C_i, C_j)}{\sqrt{\text{Var}(C_i)\text{Var}(C_j)}} \quad , \tag{5.9}$$

where Cov stands for *covariance* and Var for *variance*. By averaging this correlation coefficient over N_{pairs} randomly picked pairs of neurons, we obtain the estimator CC_{Sync} for the synchrony of population activity in a given experiment.

4. The irregularity of individual spike trains.
It is determined by the squared coefficient of variation CV^2 of the inter-spike interval distribution F_{ISI},

$$\text{CV}^2(F_{\text{ISI}}) = \frac{\text{Var}(F_{\text{ISI}})}{\text{E}^2(F_{\text{ISI}})} \quad . \tag{5.10}$$

Here, E denotes the *expectation value*. Low values of $\text{CV}^2(F_{\text{ISI}})$ indicate regular spiking, while a large value denotes irregular, Poisson-like behavior. The average CV^2 across all recorded spike trains is used to describe the network irregularity.

6 Experiments

In previous chapters of this thesis, the FACETS Stage 1 hardware system (`FHW-1`, see Section 2.1) and its operation software (Section 3.2) have been introduced. Critical problems have been reported (Section 4.3), and for some of them, methods have been presented which minimize the impact or even avoid the issues (Section 5.1). Calibration routines have been described which do not only improve the homogeneity of the silicon neural substrate, but which also establish a gauging of hardware neurons and synapses with the software simulator NEST (Section 5.2). Techniques have been presented which allow to give the hardware output a biological interpretation (Sections 3.1.5 and 3.1.6), methods were described which provide the access to variables that cannot be recorded directly (Section 4.1), and measures for the analysis of large amounts of experimental data in a biological context have been introduced (Section 5.3).

The following chapter presents both basic and complex experiments which utilize the full composition of hardware, software, optimization and data acquisition methods. These experiments represent the first steps taken on a new terrain, and many of the insights presented in previous sections have been won by early unsuccessful trials of what will be shown.

Section 6.1 comprises a set of basic studies on the single cell level. Some of them provide specifications of certain hardware features, others prove the functionality of specific hardware mechanisms or of the hardware-software interplay. Section 6.2 presents experiments on the network level, which, as far as the prototypic character and the remaining malfunctions of the utilized hardware systems allow, orient towards a utilization of the system for neuroscientific modeling.

6.1 Basic Studies and Specifications

In the following, experiments are presented which put certain components of the FACETS Stage 1 system (`FHW-1`) to the test. In Section 6.1.1, the precision of the on-chip conversion between the digital and the analog time domain is analyzed. In Section 6.1.2, a basic single neuron stimulation experiment is presented, which experimentally relates the input to the output firing rate of a hardware neuron. In addition to this pure spike analysis, Section 6.1.3 also acquires and evaluates the membrane potential traces of stimulated neurons. Sections 6.1.4 and 6.1.5 provide a proof of functionality of the FHW-1 short-term and the long-term synaptic plasticity features, respectively.

6.1.1 Spike Delivery Precision

The work presented in this section has been performed in cooperation with Dr. Andreas Grübl.

Precision of Digital-To-Time Conversion

One component of the digital controller circuitry in the FHW-1 chips is the so-called *Digital-to-Time Converter* (DTC). It performs the translation of digitally coded event packets into signals for the analog part of the chip, where no chip clock for synchronization is present. In this section two experiments are presented which analyze the temporal precision of this transition between the digital and the analog chip domain.

Measure for Conversion Quality For all kinds of conversion from digital to analog values, the so-called *differential nonlinearity* (DNL) is a commonly used measure for the transformation quality. If the digital value changes by one *least significant bit* (LSB), an ideal digital-to-analog converter changes also its analog output value by exactly the analog correspondence to this amount. The DNL measure is defined as the deviation of the analog output value changes from the ideal 1.0 LSB change (Geiger et al., 1990).

Measurement Setup As explained in Section 4.3.9, two clocks are involved in the FHW-1 digital-to-time conversion of spike events: One *chip clock*, typically operating at a frequency of $f_{\text{chip}} = 200\,\text{MHz}$ (HTD), and a so-called *bin clock*, which generates 16 sub-bins for each clock period T_{chip} of the chip clock. One period T_{bin} of the bin clock corresponds to the LSB of the DTC process. Hence, an ideal temporal resolution of the digital-to-time event transformation at a chip operation frequency of $f_{\text{chip}} = 200\,\text{MHz}$ is $T_{\text{res}} = 1/(16 \cdot 200\,\text{MHz}) = 312.5\,\text{ps}$ (HTD). But due to imperfections of the conversion process, its DNL cannot be expected to be zero. The dominating source of imperfection can be identified by the following experiment.

Since the precision measurement shall cover both clocks involved in the event generation process, the data acquisition covers four cycles of the chip clock, which equals 64 cycles of the bin clock. For one specific synapse driver per FHW-1 chip, the voltage peak which represents the event in the analog domain of the chip (see Section 2.1.2) can be recorded via the multiplexed output pin described in Section 2.1.5. Two events are delivered to this synapse driver, with a time difference of 6401 bin clock cycles or 400 chip clock cycles plus 1 LSB of the DTC. The chip clock is set to $f_{\text{chip}} = 200\,\text{MHz}$ (HTD). Then, for 64 times, both events are shifted in time by one LSB of the DTC process, i.e. their time difference in the *digital* domain does not change, but an increasing start offset of t_{offset} is applied. For every run, the time difference Δt_{osc} between the resulting *analog* voltage peaks is recorded with an oscilloscope via the multiplexed pin.

Results Figure 6.1 shows the difference between the acquired time differences Δt_{osc} and the ideal value Δt_{theo} of 6401 time bins as a function of t_{offset}. Every data point represents the mean value over 100 measurements, while the error-bars indicate the standard errors of the mean values. A systematic pattern can be observed, which clearly dominates the statistical error: Every time t_{offset} has been increased by 16 bins, Δt_{osc} drops significantly below the ideal value. One time later, it increases significantly above the ideal value. This effect results in a quality measure for the DTC process of $\text{DNL}_{\text{DTC}} \approx 0.7$ LSB. For a chip frequency of $f_{\text{chip}} = 200\,\text{MHz}$, this corresponds to a temporal precision of the DTC process of $T_{\text{res}} \approx (300 \pm 200)$ ps (`HTD`).

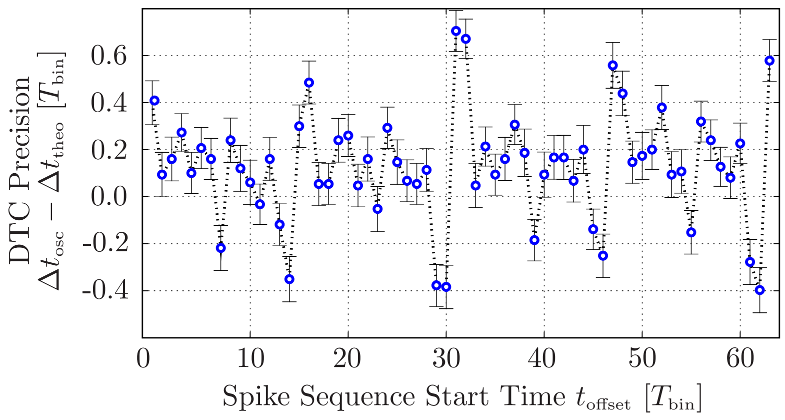


Figure 6.1: Deviation of the time interval Δt_{osc} between two spikes generated by the `FHW-1` DTC logic (acquired with an oscilloscope) from its corresponding ideal value Δt_{theo}, as a function of the start offset t_{offset} with which the two spikes are fed into the chip. Every data point represents the mean value over 100 measurements, the error-bars indicate the standard errors of the mean values.

Analysis The reason for the systematic peaks becomes clear from the following additional experiment: The same synapse driver as before is fed with 16 events at times t_i. The time t_0 of the first spike is chosen such that the sub-bin index into which it is sorted is zero: $t_0 = N \cdot T_{\text{chip}}$, with N being a natural number. The time difference between two subsequent spikes is $t_{i+1} - t_i = M \cdot T_{\text{chip}} + 1 \cdot T_{\text{bin}}$, where M is a natural number, i.e. the sub-bin index into which the events are arranged increases by one for every event. The analog event representation is recorded with the oscilloscope, and so is a membrane potential which receives excitatory conductance changes from the observed synapse driver via a strong synapse node (see Section 2.1.2). The resulting traces are plotted in Figure 6.2. The lower trace shows the voltage pulses generated by the DTC circuitry which represent the events in the analog chip domain. The upper trace shows the membrane potential of the stimulated neuron. More precisely, both traces show an average over multiple repetitions of the experiment.

The analog pre-synapse-driver event signal decreases with a growing time bin, which is to be expected due to the fact that its amplitude is generated from the length of an enable signal which lasts from the onset of the event until the end of the corresponding chip clock cycle. Hence, the higher the time bin within that cycle, the shorter the enable signal and the smaller the event peak. This observation also explains the peaks in Figure 6.1: The oscilloscope acquisition of the analog event time representations involves a software peak detection algorithm. These peaks are low-pass filtered by the readout RC-network, and larger amplitudes in the original analog signal result in an earlier detection due to larger slopes in the filtered signal. Hence, the measured analog time is distorted by a systematic offset that depends on the time bin of its original digital representation. This results in

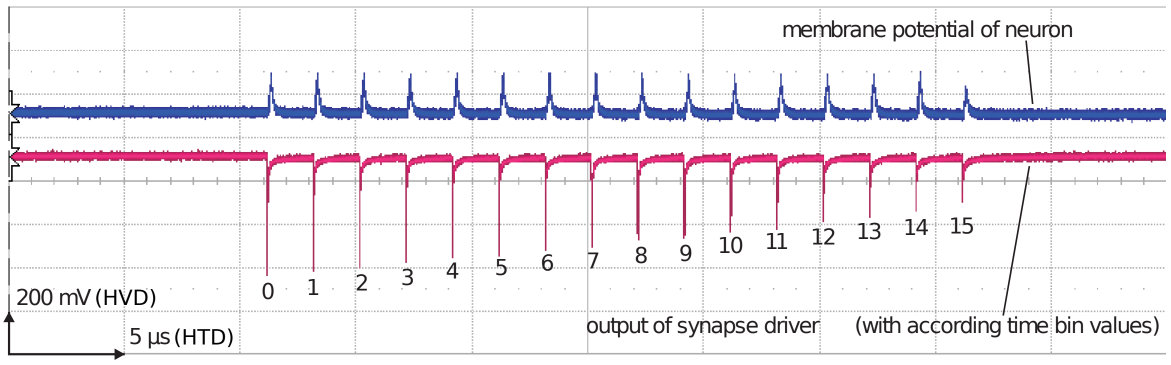


Figure 6.2: Lower trace: An oscilloscope measurement of voltage pulses generated by the DTC circuitry, which represent a sequence of equidistant spike events in the analog domain of a `FHW-1` chip. The pulses have been generated with different time stamps, such that every value of the bin clock between 0 and 15 is met once. As expected from the design, the amplitudes of these signals decrease with growing time bins. Upper trace: A membrane potential stimulated by these events via an excitatory synapse. Both plotted traces represent an average over multiple runs. The smaller amplitude of the last post-synaptic potential in the upper trace is explained in the main text. Figure by A. Grübl.

the observed jump of the measured value Δt_{osc} from negative to positive deviations from its target value every 16 time bins, i.e. when the difference between the amplitudes of the analog representations of the two spikes rapidly changes.

Observation of a Chip Malfunction But Figure 6.2, which has been acquired with an `FHW-1.2` chip, indicates another problem, which has been already indicated in Section 4.3.9. Due to the small amplitudes of analog event representations corresponding to the last time bin within one chip clock cycle, these events sometimes do not initiate a conductance course generation at the connected synapse driver. The amplitude, also low-pass filtered by on-chip wiring, is just not sufficient to exceed a necessary trigger level. This problem is caused by a disadvantageous wire routing automatedly performed by a chip design software. The issue has been understood and described in (Grübl, 2007, Section 6.5.4), and is solved for the third generation of the `FHW-1` chips.

Precision of Time-To-Digital Conversion

After the translation of event times from the digital to the analog chip domain has been analyzed, the opposite direction is tested. The goal is to measure the accuracy of the digitalization of events generated in the analog part of the chip (time-to-digital conversion, TDC).

Measurement Setup Again, for a specific synapse driver, the incoming voltage peaks which represent the event times are recorded with an oscilloscope. Again, two events arrive at this driver, and the temporal distance Δt_{osc} between them is acquired. In contrast to the DTC setup, though, these events are not directly generated by an externally applied spike signal, but by the event output of a hardware neuron which feeds its output into the observed synapse driver. The delivering neuron itself receives two sets of four synchronous spikes each via strong excitatory synapses. This results in two output spike generated by the stimulated neuron, which are not only fed into the synapse driver, but are also digitalized by the on-chip TDC logic. The time difference Δt_{out} between these digital output time stamps and the reference measurement Δt_{osc} is computed. This procedure is repeated for different temporal offsets t_{offset} of the applied stimulation spike sets, while the period between the two sets always stays the same.

Results Figure 6.3 shows the acquired results. Due to problems in the on-chip event capturing logic, which are described in detail in (Grübl, 2007, Section 4.3.5), the accuracy check is only performed on the scale of a chip clock period T_{chip}.

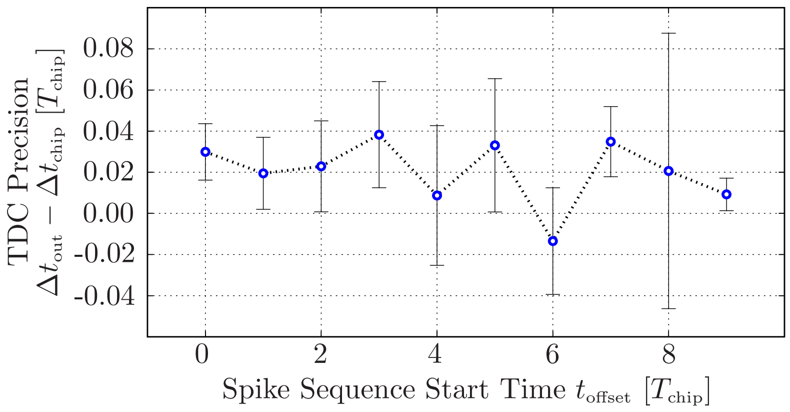


Figure 6.3: Deviation of the time interval Δt_{out} between two spikes digitalized by the `FHW-1` TDC logic and a reference measurement Δt_{osc}, as a function of the start offset t_{offset} with which the two spikes are fed into the chip.

For the plotted ten different stimulation pattern offsets, no deviation of the digitalized time difference Δt_{out} from the analog reference measurement Δt_{osc} is larger than $0.05 \cdot T_{\text{chip}}$. This is considered to prove the functionality of the time-to-digital event conversion on a chip clock cycle basis.

6.1.2 Firing Rates

In the following, a basic single-neuron experiment is set up. One cell is stimulated by a set of Poisson-type spike trains, and the resulting output firing rate as a function of the applied input rates is analyzed. This is done both in hardware and with the software simulator NEST, utilizing the same PyNN scripts. The goals of the presented experiments are

- to give a basic example of how to use the simulator-independent modeling language PyNN by means of a line-by-line code explanation,
- to prove the functionality of the `FHW-1`-specific PyNN implementation and the resulting portability of experiments between the hardware and software simulators as motivated in Section 3.1.3,
- to compare the results acquired from both back-ends, to discuss the achievements and remaining shortcomings of the matching,

- to experimentally demonstrate the hardware-specific input bandwidth limitations described in Section 4.3.7.

Input vs. Output Rates in Hardware and Software

A first experiment focuses on the matching between an `FHW-1` and a NEST neuron in terms of output firing rates under the same stimulation. One type of hardware input bandwidth limitation can be observed already in this case, though.

Setup In a first setup, a single neuron is created, and $N_\mathrm{e} = 48$ excitatory plus $N_\mathrm{i} = 16$ inhibitory Poisson-type spike trains are connected to it. All stimulation spike trains fire with an adjustable frequency f_stim for a period of $T_\mathrm{exp} = 5000\,\mathrm{ms}$ (`BTD`). The maximum synaptic conductance g^max is 2.0 nS for excitatory and 15.0 nS (`BVD`) for inhibitory connections. The output spikes of the stimulated neuron are recorded. This experiment is fully described in PyNN and executed both on the `FHW-1` system and with NEST.

PyNN Description, Step by Step Listing 6.1 shows the PyNN script which describes the experiment. In line 1, the simulation or emulation back-end, in this case the `FHW-1` system, is chosen. In order to utilize NEST instead, the only necessary change within this script is to replace line 1 by `from pyNN.nest2 import *`, everything else remains the same. From line 4 to 8, the numbers of external stimuli and the synaptic weights are set. In lines 10 to 16, the neuron parameters are defined. Lines 19 and 20 determine the rate and duration of the Poisson spike train stimuli. In line 23, PyNN is initialized, the numerical integration step size of 0.1 ms is passed. If the hardware back-end is chosen, no discrete step size is utilized due to the time continuous dynamics in its analog network core. Instead, the function argument is used to determine the time resolution for the membrane potential sampling via the oscilloscope, if connected. In line 25, the neuron is created, passing the cell parameters and the number of cells as the second and the third arguments. The first argument, `IF_facets_hardware1`, specifies the neuron type to be created. For the `FHW-1` system, no other neuron type is available. For the NEST back-end, this neuron type determines parameter values for e.g. the membrane capacitance C_m, which are fixed to resemble the hardware.

In lines 27 and 28, the Poisson-type spike sources are generated, passing the type of source, the previously defined parameters and the desired number. In lines 30 and 31, the spike generators are connected to the neuron. The arguments of the connect commands first specify a list of sources, then a list of targets, followed by the synaptic weights, the synapse types and finally by the probability with which each possible pairing of source and target objects is actually connected. The recording of the output spikes and the membrane potential of the stimulated neuron is prepared in lines 34 and 35. In line 37, the experiment is executed for a duration of 5000 ms (`BTD`). Line 40 defines the end of the script, and initiates the writing of the recorded values to files.

Listing 6.1: PyNN description of a single-neuron stimulation experiment. For a detailed explanation see main text.

```
from pyNN.hardware.stage1 import *
# OR: from pyNN.nest2 import *

numExcInputs =  48
numInhInputs =  16

wExc = 0.002  # uS
wInh = 0.015  # uS

neuronParams =    { 'v_reset'    : -80.0,    # mV
                    'e_rev_I'    : -80.0,    # mV
                    'v_rest'     : -70.0,    # mV
                    'v_thresh'   : -55.0,    # mV
                    'g_leak'     :  40.0,    # nS
                    'tau_syn_E'  :  30.0,    # ms
                    'tau_syn_I'  :  30.0,    # ms
                  }

inputParameters = { 'rate'       : 5.0,      # Hz
                    'duration'   : 5000      # ms
                  }

setup(timestep=0.1)

neuron = create(IF_facets_hardware1,neuronParams,n=1)

iExc = create(SpikeSourcePoisson,inputParameters,n=numExcInputs)
iInh = create(SpikeSourcePoisson,inputParameters,n=numInhInputs)

connect(iExc,neuron,weight=wExc,synapse_type='excitatory',p=1.0)
connect(iInh,neuron,weight=wInh,synapse_type='inhibitory',p=1.0)

record(neuron, 'spikes.dat')
record_v(neuron, 'membrane.dat')

run(5000)  # duration in ms
end()
```

Results The experiment is run both on the FHW-1 system and with the software simulator NEST. The firing rate of the stimulating Poisson spike trains was varied from 1 Hz to 20 Hz (BTD) in steps of 1 Hz, and for each rate the experiment was repeated 30 times with different random number generator seeds. Figure 6.4 shows the resulting average output firing rates.

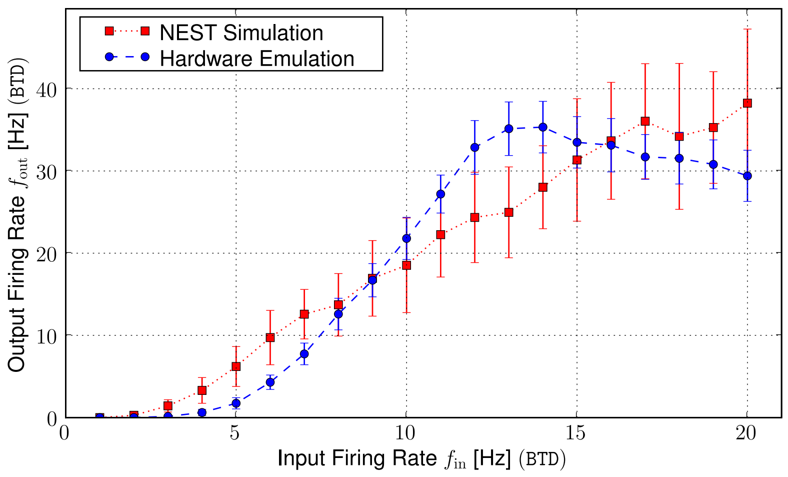


Figure 6.4: A single neuron is stimulated by Poisson-type spike trains via 48 excitatory and 16 inhibitory synapses. The plot shows the average output firing rate of the stimulated neuron as a function of the applied input rates. The setup (see PyNN description in Listing 6.1) has been executed with various stimulation rates on both the `FHW-1` system (circles) and the software simulator NEST (squares). Each data point represents the mean over 30 runs, the error-bars denote the corresponding standard deviations. The drop of the hardware output rate for input frequencies larger than approximately 12 Hz (`BTD`) is a result of limited buffer depths in the memory-to-chip event transmission as described in Section 4.3.7. Therefore, above a critical input rate, not all input-spikes actually reach the hardware anymore. Further discrepancies between the results of both back-ends are discussed in the main text.

Analysis For input firing rates smaller than approximately 12 Hz (`BTD`), the output firing rates acquired from both back-ends exhibit a coarse qualitative correspondence. As intuitively expected, from a certain critical input rate, on both back-ends the output firing rate grows with the input rate. For the hardware system, though, the onset of activity is later, and the following slope is steeper. The best matching is achieved for input rates of (9 ± 1) Hz (`BTD`), where both curves meet, which can be explained by the fact that for this output frequency, the synaptic weight calibration factors (see Section 5.2.5) have been determined. The delayed onset of the hardware activity and the seemingly growing responsiveness for higher input rates is assumed to be an effect of the parasitic capacitances incorporated in the synaptic transmission path in hardware (see Sections 4.3.4 and 6.1.4). The resulting damping effect decreases for higher signal frequencies, because then these capacitances have less time to lose their accumulated charge. Hence, a perfect `FHW-1`-NEST alignment of such f_{in} vs. f_{out} curves is not possible, a matching can only be achieved for a narrow range of input frequencies.

For input firing rates larger than approximately 12 Hz (`BTD`), the communication bandwidth between the event memory module and the analog part of the chip is limited by the depth of the involved FIFO buffers. This issue has been described in Section 4.3.7, and becomes very obvious in the presented experiment. When the buffers overflow, not all applied input spikes can be delivered into the analog part of the chip anymore (see also Section 6.1.1), which results in a stagnancy of the output firing rate. The fact that the output rate in Figure 6.4 even decreases for rates larger than approximately 13 Hz (`BTD`) is possibly caused by a higher loss of excitatory input events compared to the loss of inhibitory ones.

Experimental Demonstration of Another Bandwidth Limitation

Setup Like in the previous setup, a single neuron is created, but this time $N_\mathrm{e} = 208$ excitatory plus $N_\mathrm{i} = 48$ inhibitory Poisson-type spike trains are connected to it. This exploits the full number of synapse drivers available for one `FHW-1` neuron, and for spike trains with identical firing rates, the maximum input bandwidth in this case is not determined by the FIFO buffer depths, but by the event packet transmission limit of 300 MEvents/s (`HTD`), i.e. by the chip clock frequency (see Section 4.3.4). This time, the maximum synaptic conductance g^max is 1.0 nS for excitatory and 15.0 nS (`BVD`) for inhibitory connections, which results in reasonable output firing rates.

Results Figure 6.5 shows the resulting f_in vs. f_out curve. As predicted in Section 4.3.7, the input event data cannot be reliably delivered to the hardware neuron from a critical rate per spike train of approximately $f_\mathrm{in} \approx 8$ Hz (`BTD`). This input bandwidth limitation has to be considered during the setup of every hardware experiment.

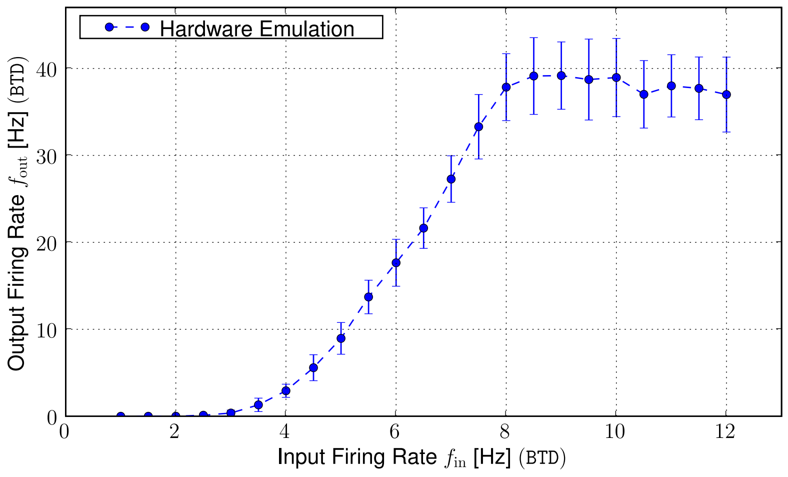


Figure 6.5: A single `FHW-1` neuron is stimulated by Poisson-type spike trains via 208 excitatory and 48 inhibitory synapses. The plot shows the average output firing rate of the stimulated neuron as a function of the applied input rates. Each data point represents the mean over 30 runs, the error-bars denote the corresponding standard deviations. The stagnancy of the hardware output rate for input frequencies larger than approximately 8 Hz (`BTD`) is a result of the limited memory-to-chip event packet transmission bandwidth as described in Section 4.3.7. I.e. from a certain input rate, not all applied spikes reach the hardware.

6.1.3 Membrane Potentials

In the previous section, only average output firing rates have been considered as measures of neuronal activity. But since every spike is the consequence of evolving membrane potential dynamics (see Section 2.1.2), and since the firing rate is not the only variable used by neurons to code output information into spikes (see e.g. Thorpe et al., 2001), acquiring the sub-threshold development of neuron membrane voltages can provide many additional insights. Especially in the case of neuromorphic hardware systems, where the correspondence between the electronically emulated membrane potential and its biological counterpart has to be established by applying dedicated translation and calibration methods (as presented in Sections 3.1, the analysis of membrane potentials is essential for the verification of the model.

The goals of the experiments presented in the following are

- to prove the functionality of the `FHW-1`-specific PyNN implementation in terms of membrane potential data access, i.e. to demonstrate the automated integration and translation of oscilloscope data into the PyNN domain,
- to compare the hardware voltage traces directly with corresponding NEST simulation data (see Section 3.1.4),
- to illustrate the improvements achieved by the calibration methods introduced in Section 5.2 in terms of matching between the hardware system and NEST.

Setup A single neuron is created and stimulated with $N_{\mathrm{e}} = 48$ Poisson-type spike trains via excitatory plus $N_{\mathrm{i}} = 16$ spike trains via inhibitory synapses. All stimulation spike trains fire with a frequency $f_{\mathrm{stim}} = 8\,\mathrm{Hz}$ (`BTD`) for the full experiment duration of $T_{\mathrm{exp}} = 5000\,\mathrm{ms}$ (`BTD`). This stimulation frequency value has been chosen, because it results in an optimal matching of the hardware and the NEST output firing rates, as has been shown in Section 6.1.2. The maximum synaptic conductance g^{max} is set to $2.0\,\mathrm{nS}$ for excitatory and to $15.0\,\mathrm{nS}$ (`BVD`) for inhibitory connections. The membrane potential and the output spikes of the stimulated neuron are recorded. The experiment is fully described in PyNN and executed both on the fully calibrated `FHW-1.3-No.18` system and with NEST.

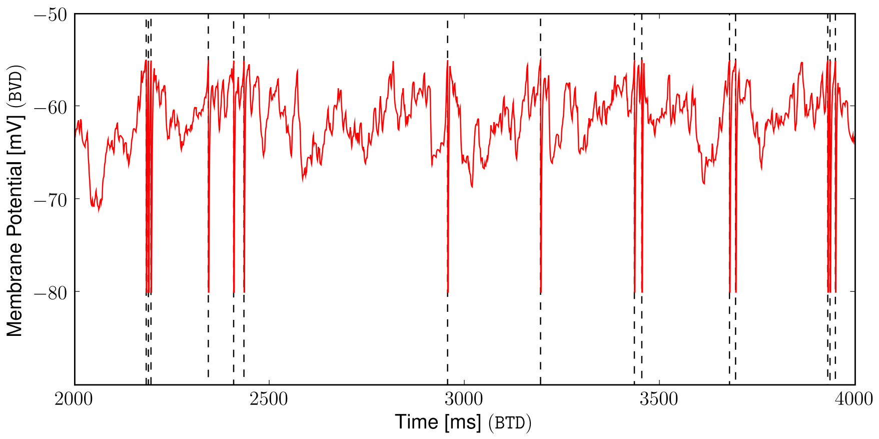
For the `FHW-1` back-end, the same setup is repeated 50 times. The hardware run which matches the NEST result best in terms of the spike train difference measure introduced in Section 5.3.1 (with the parameter τ_c set to $3\,\mathrm{ms}$, `BTD`) is selected for the membrane potential comparison shown in Figure 6.6.

In further hardware runs, the system fine-tuning is deactivated one by one in terms of

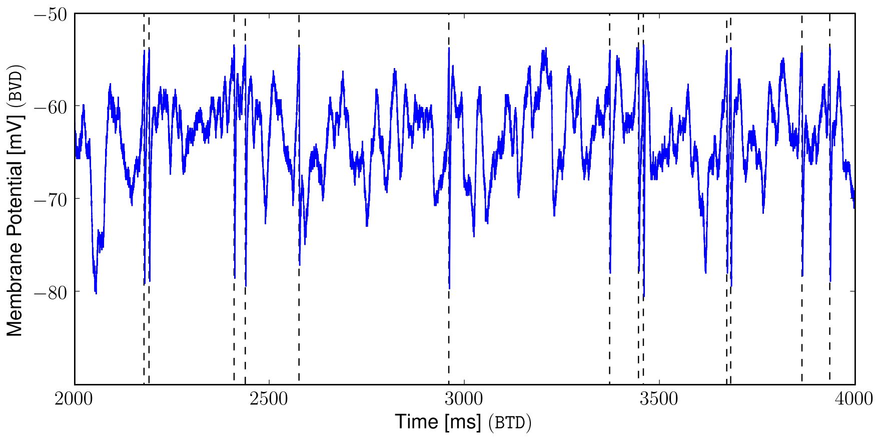
- the synapse weights calibration (see Section 5.2.5),
- the membrane time constant calibration (see Section 5.2.3),
- the firing threshold and reset mechanism calibration (see Section 5.2.2),
- the synapse driver calibration (see Section 5.2.4).

The resulting changes on a membrane potential exposed to the same input as described above illustrate the improvements achieved by each calibration method.

Results For the period between $2000\,\mathrm{ms}$ and $4000\,\mathrm{ms}$ (`BTD`), Figure 6.6 shows the membrane potential and the spike times (indicated by vertical dashed lines) of the stimulated neuron as computed in a NEST simulation (upper sub-figure) and as emulated by the `FHW-1` system (lower sub-figure). There is an obvious qualitative correspondence between both traces, and 10 out of the 15 plotted spike times generated by NEST are reproduced on the hardware neuron (temporal offsets in the order of $10\,\mathrm{ms}$ `BTD`are tolerated). The matching is not perfect,



(a) NEST Simulation



(b) Hardware Emulation

Figure 6.6: The sub-threshold membrane potential of a neuron stimulated by Poisson-type spike trains as simulated by NEST **(a)** and emulated by the `FHW-1` system **(b)**. Both in NEST and in the hardware model, action potentials are not modeled with depolarization peaks, but just registered in terms of their occurrence time and followed by a reset mechanism. Therefore, the output spike times of the recorded neuron are indicated by vertical dashed lines. The voltage traces show an extract of 2000 ms (`BTD`) length out of the full experiment duration of 5000 ms (`BTD`).

though. The hardware does not reproduce all NEST spikes, and generates others, which are not part of the NEST model. The hardware membrane trace seems to be subject to stronger fluctuation amplitudes. Especially in the case of polarizing peaks in the hardware trace, the amplitudes are often much larger than corresponding drops of the NEST voltage. Different reasons for this can be assumed: The efficacy of one or multiple of the involved synapse drivers have not been sufficiently calibrated by the synapse driver calibration routine presented in Section 5.2.4 – they are still too strong. There is not yet a mechanism incorporated in the utilized software framework which avoids such outliers in the process of mapping biological networks to the available hardware substrate, as it is already used for critical hardware neurons (see Section 3.1.6). Another possible reason is a particularly large sensitivity of the circuit which translates the synaptic conductance course signal (chip-internally represented as a current, see Section 2.1.2) into a conductance between the membrane and the inhibitory reversal potential. This can be caused by transistor-level fluctuations specific to the observed neuron.

Deactivating Calibration Mechanisms The experimental setup is kept, but in four separate runs, each time one calibration method is deactivated. For illustrative purposes, a neuron has been manually selected for which these disabling steps have an impact that can be easily observed on the membrane.

Figure 6.7 shows six membrane potential traces, again for the period between 2000 ms and 4000 ms (`BTD`) of experiment time. The upper-most trace represents the NEST simulation. Trace A is recorded during a run with all calibration mechanisms presented in Section 5.2 applied.

For the recording of trace B, acquired with the same neuron as in case A, the synapse weight calibration presented in Section 5.2.5 has been deactivated. This results in a significantly increased output firing rate, caused by larger excitatory synaptic weights. The correspondence with the reference NEST model decreases.

For the recording of trace C, the only disabled calibration mechanism is the tuning of the membrane time constant (see Section 5.2.3). In this case, the applied default value results in a membrane time constant which is too low, i.e. the impact of synaptic stimulation is not enough to drive the membrane potential above the firing threshold. This also decreases the correspondence with the reference NEST model.

Trace D has been acquired from the same neuron, but with the calibration mechanism deactivated that determines the trade-off between firing threshold precision and reset mechanism efficacy (see Section 5.2.2 – the necessity of such a trade-off is a design-related malfunction of the `FHW-1` system). The resulting trace looks like it fits the NEST reference better than trace A, because the reset mechanism seems to work more efficiently, and the spike times of the NEST model are better reproduced. But the originally applied calibration algorithm has detected a responsiveness of this neuron which is too high without further tuning, and although this cannot obviously be seen from trace D, the calibration result is trusted.

For the recording of trace E, the synapse driver calibration (see Section 5.2.4) has been deactivated. The applied default values result in much weaker synaptic impacts compared to the calibrated case. Hence, the membrane fluctuation amplitudes are very small, and consequently, the neuron does not fire a single spike. Better default values would improve this shortcoming, but finding those values requires additional effort. Investing this effort has been considered to be not reasonable, because values, which are tuned for each individual driver, are already available.

6.1.4 Short-Term Plasticity

In Section 2.1.2, the synaptic depression and facilitation mechanisms implemented in the `FHW-1` system have been described. Their functionality is illustrated by the following experiment. The data presented in this section has been acquired by Johannes Bill under the supervision of the author (see also Bill, 2008).

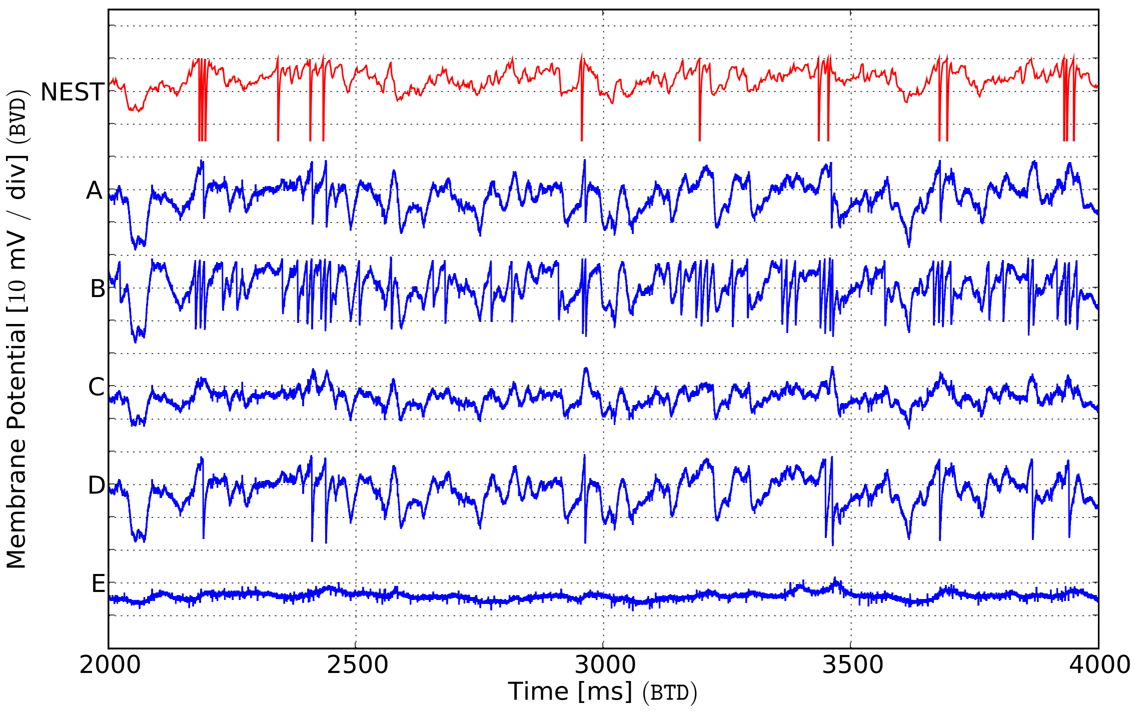


Figure 6.7: The sub-threshold membrane potential of a neuron stimulated by Poisson-type spike trains as simulated by NEST (top-most trace), as emulated by a fully calibrated `FHW-1` neuron (A), by the same neuron without synapse weight calibration (B), without membrane time constant calibration (C), without firing threshold vs. reset mechanism tuning (D), and without synapse driver calibration (E). For each trace B – E, only *one* calibration mechanism has been deactivated. See main text for more explanations on the observable differences between the traces.

Setup The short-term dynamics of one specific, excitatory synapse shall be analyzed. Therefore, a single neuron is exposed to 16 Poisson-type spike trains via excitatory and 4 Poisson-type spike trains via inhibitory synapses, each of which fires with a rate of 5 Hz (`BTD`). In addition to this background input, the stimulated neuron also receives a specifically prepared spike train via the synapse of interest: 30 spikes with constant inter-spike intervals of 50 ms (`BTD`), followed by one extra spike 500 ms (`BTD`) later, are injected. The resulting membrane potential is recorded over 100 repetitions of this setup, each time with new randomly generated Poisson spike trains applied.

These 100 runs are performed three times: Once, the plasticity mechanism in the analyzed synapse is configured to be facilitating, once it is switched off (static synapses) and once it is depressing. The workaround presented in Section 5.1.4 is applied in order to minimize design-related problems. See Bill, 2008, Section I.3.3 for the precise hardware parameters applied.

Results Figure 6.8 shows the recorded membrane potentials. It is divided into three sub-figures, the upper one showing the facilitating synapse, the middle one showing the case without short-term plasticity, and the lower one showing the depressing synapse. In each of these sub-figures, two traces are plotted: The upper of both represents the average membrane potential over all acquired 100 runs, while the lower one represents one out of these 100 measurements.

While the membrane potential fluctuations imposed by the Poisson-type background stimulation dominate the single run traces, they are filtered out in the average traces. There, the three different characteristics of the synapses can clearly be observed: In the facilitating case, the synapse needs approximately 5 μs (`HTD`) (or 500 ms in `BTD`) of permanent stimulation before it reaches its maximum efficacy. Approximately the same period is needed by the depressing synapse, before it has reached its minimum impact. In both cases, the post-synaptic potential caused by the delayed extra spike illustrates the recovering of the synapse from the depression or facilitation effect.

An interesting observation can be made in the middle sub-figure. The synapse, which is configured to exhibit no plasticity mechanism at all, seems to need a certain amount of stimulation before it reaches its maximum efficacy, just like for a facilitating configuration. This is interpreted as an effect of parasitic capacitances within the synaptic circuitry, which need to be pre-charged before the synaptic conductance courses reach their full amplitude (see Section 2.1.2 for an explanation of the affected electronic circuits).

Still, despite the need for workarounds to be applied and despite parasitic effects, the basic principles of synaptic depression and facilitation can be stated to be working in the analyzed `FHW-1` synapse. See Section 6.2.2 for an experiment which utilizes the synaptic depression and facilitation for self-stabilizing dynamics in a recurrent neural network.

6.1.5 Long-Term Plasticity

In Section 4.1.3, a method has been presented which allows to access the STDP modification curves of hardware synapse nodes. The Figure 6.9 shows a set of STDP curves acquired with this method from adjacent synapses on the same `FHW-1.3-No.17` chip. Every data point represents the mean value, the error bars represent the standard deviations over 10 repetitions of the same acquisition setup. As mentioned in Section 4.3.11, technical problems with the controlling of these curves are still unsolved. Hence, it is not clear if the observed synapse-to-synapse differences and the error bars within one curve are caused by actually varying hardware behavior or by problems in the acquisition process. Possible distorting factors are listed in Section 4.3.11. Concluding statements about the measurements can therefore not be given at this point.

Despite the scepticism regarding the quality of the results, the presented STDP curve array illustrates the ability of every synapse to measure temporal correlations between pre- and post-synaptic spiking activity. The time constants of the measured branches, which are not adjustable by a hardware parameter, are in the order of 5 ms to 10 ms (`BTD`). This is rather short compared to choices of around 20 ms (`BTD`) found e.g. in (Song et al., 2000; Morrison et al., 2007).

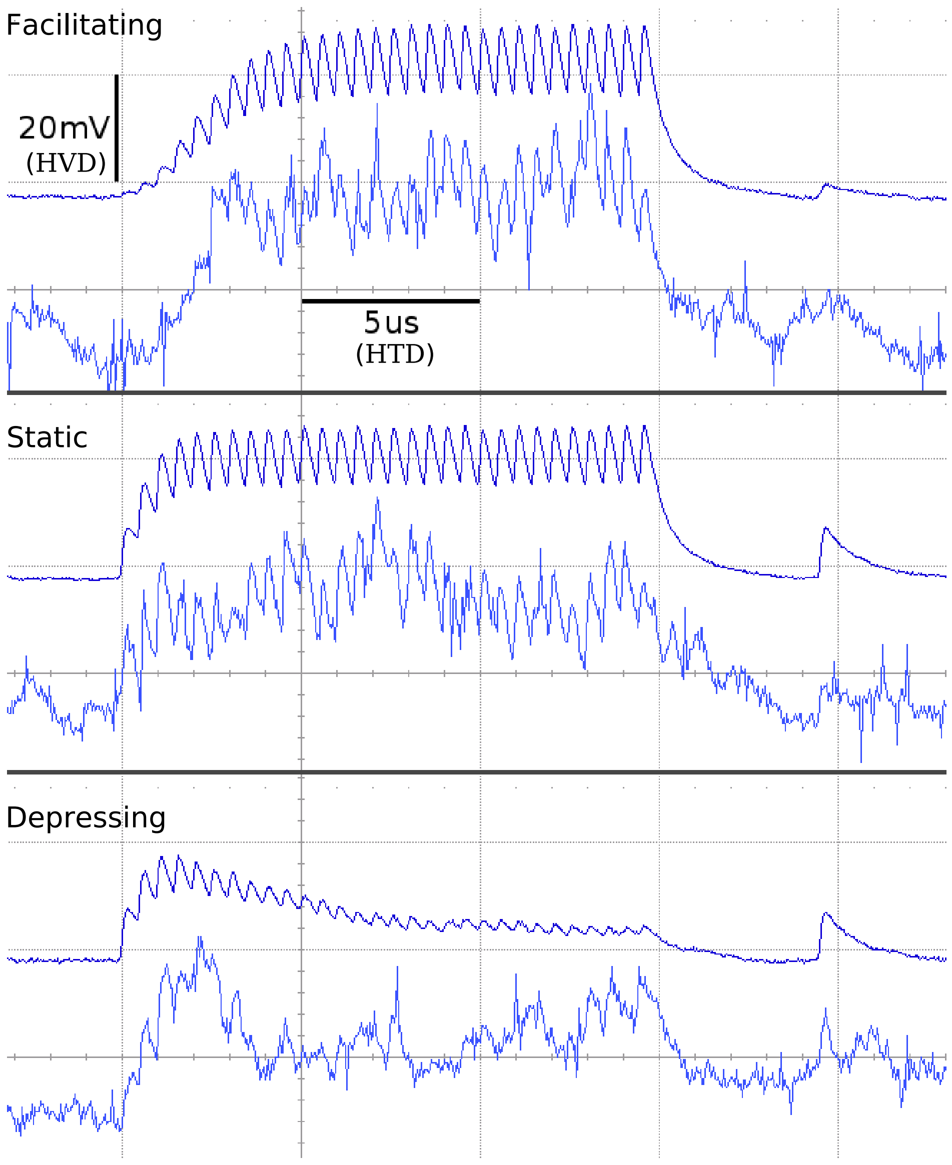


Figure 6.8: A hardware membrane potential exposed to Poisson-type background stimulation and to equidistant spikes via one *facilitating* (top sub-figure), one non-plastic (middle) and one *depressing* (bottom) synapse. In every sub-figure, an average over 100 recordings of the membrane with newly generated background stimuli is shown (upper trace), plus one single recording out of these 100 measurements (lower trace). For the single-run membrane potentials, the fluctuation due to the background stimulation dominates the dynamics. In the average traces the background is filtered out, and the impact of the analyzed synapse becomes visible. In the facilitating case, the impact of the synapse grows with ongoing stimulation, until a maximum efficacy is reached. The opposite effect can be observed for the depressing synapse. Both effects recover in the absence of ongoing stimulation, as can be seen from the effect of a delayed single spike. The synapse which is configured to exhibit no plasticity at all still needs a certain amount of stimulation before it reaches its maximum efficacy, which is assumed to be caused by parasitic capacitances in the synapse circuitry. Figure by J. Bill.

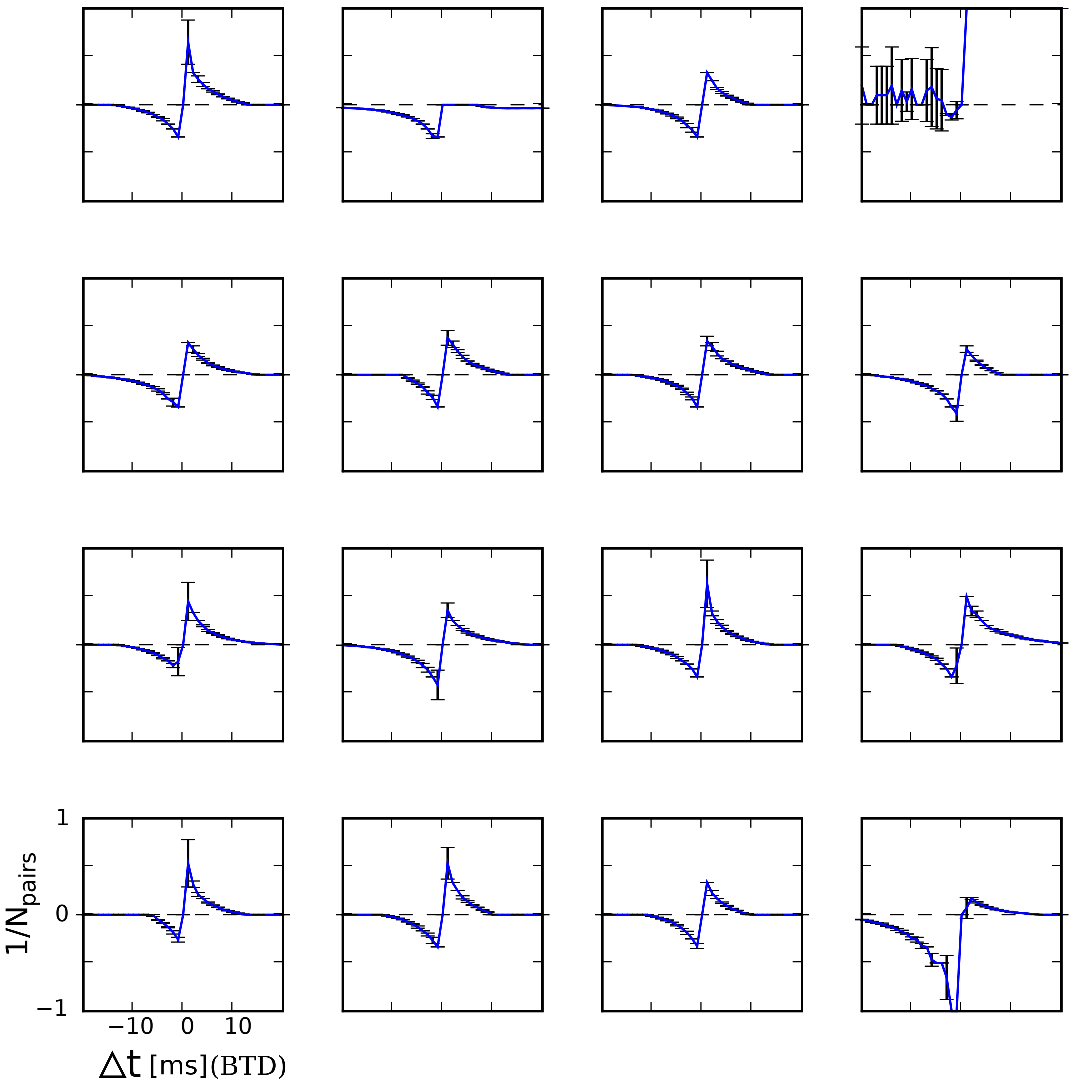


Figure 6.9: An array of STDP modification curves measured in 16 adjacent synapse nodes that are connected to the same `FHW-1.3-No.17` neuron. The corresponding measurement method is explained in Section 4.1.3. Pairings of pre- and post-synaptic spikes with a time difference of Δt are artificially provoked in the synapse node. The number N_{pairs} of these pre- / post-synaptic spike pairs that has to be applied until a hardware-internal flag signals a necessary weight update is determined. The inverse of $N_{\mathrm{pairs}}(\Delta t)$ is plotted against Δt. Every data point represents the mean value, the error bars represent the standard deviations over 10 repetitions of the acquisition setup.

6.2 Exploring Network Architectures

In this section, experiments with basic neural network architectures implemented on the chip-based FACETS hardware system (`FHW-1`, see Section 2.1) are presented. The applied setups are described, executed and analyzed with the software framework introduced in Section 3.2. They incorporate the calibration methods and the chip malfunction handling techniques described in Sections 5.2 and 5.1.

This section points out the large effort that has to be invested in order to realize even simple functional neural architectures on a neuromorphic hardware system. But it also provides the proof that a realization *is* possible. And that, for certain experiments, the advertised benefits in terms of operation speed already pay off for the utilized prototype device.

The following studies address different aspects of cortical neural networks: In Section 6.2.1, a recurrent network inspired by cortical connectivity patterns is configured and stimulated such that a spectrum of firing dynamics emerge at least one of which is observed *in vivo*. A parallel execution on both the hardware system and a software simulator, with a mutual matching approach, provides additional insights into hardware dynamics and delivers direct computation time comparisons.

In Section 6.2.2, the synaptic short-term plasticity mechanisms implemented in the `FHW-1` system are utilized to generate self-stabilizing firing dynamics in a recurrent neural network. This approach can help to provide a neural substrate that is tolerant of certain types of hardware-specific inhomogeneities.

6.2.1 Recurrent Network Dynamics: Matching Hardware and Software

One of the conditions for the establishment of neuromorphic hardware as a tool for the modeling neuroscience community has been stated in Section 1.2.4: Its biological relevance must be proven. The major aim of the following experiment series is to show that and how this condition can be fulfilled. The `FHW-1` system is not an optimal substrate for this purpose due to its prototype nature, i.e. its limited size (see Section 2.1.1) and its various malfunctions (see Section 4.3). Nevertheless, the realization of the comparison paradigms presented in this section yields practical experience and solutions for the overcoming of emerging obstacles. The acquired insights are expected to be useful for the operation of future systems.

The chosen reference system the hardware shall be compared with is not biological tissue, though, but a software simulator. This is a reasonable method, because the hardware system implements a model which already strongly abstracts the detailed knowledge of the high diversity of cortical cell types that has been accumulated by neuro-physiologists (for a review see Toledo-Rodriguez et al., 2002). At the same time, there is a large community of scientists who contribute insights into cortical dynamics by utilizing software simulations at a comparable abstraction level (see e.g. Destexhe et al., 2003; Helias et al., 2008; Tsodyks and Markram, 1997).

One further purpose is implicitly met with the experiments documented in this section: Nearly the full set of insights and methods described previously in this thesis is unfolded and applied to one specific and biologically motivated experiment. All these deployed techniques help to achieve a homogeneous neural substrate on an `FHW-1` chip, with a minimum of distortions caused by process-inherent or design-related issues, and with an optimized matching between the hardware and the reference software model (see Section 6.1.3).

In the following, the activity of a self-stimulating, i.e. recurrent network with a simple structure of two randomly connected sub-populations under external stimulation is both emulated using the `FHW-1` hardware system and computed with the software simulator NEST (see Section 3.1.4). The statistical descriptors previously defined (see Section 5.3.2) are applied for the analysis of the generated output data. They are shown to be an appropriate tool to find hardware and software configuration regimes which make both platforms respond similarly to identical input while exhibiting biologically realistic network dynamics. Due to distortions of the hardware results caused by various issues (see Section 6.2.1), the results presented here cannot yet provide the basis for a final evaluation of this matching approach. Still, the full experiment is described in the following, including many techniques to handle the mentioned problems, and the distorted but promising results acquired with the `FHW-1.3` chip are presented. The reasons for remaining differences between the behavior of both platforms are discussed with a focus on the improvements that are to be expected from a future revised chip that solves the critical issues.

The work presented in this section has been performed in cooperation with Jens Kremkow[1]

The Cortical Architecture and its Dynamical States

Describing the architecture of cortex is important for understanding its computational principles. Early staining studies have shown that the cortex could be considered as a vertical layered structure (Ramon y Cajal, 1911), in which neurons and their projections are laminar specific (reviewed in Douglas and Martin, 2004). Studies have provided an estimate of the total number of neurons and synapses in the different layers of the cat visual cortex (Beaulieu and Colonnier, 1983, 1985). However, a consensus about how these synapses are distributed across the neurons still is not reached.

Despite the layered structure, it is argued that there is only little fine anatomical specificity within the local cortical volume and that the connectivity could be considered as random (Braitenberg and Schüz, 1991). However, technical advances have produced new data which highlights the non-randomness of the cortex (Binzegger et al., 2004; Stepanyants et al., 2008; Binzegger et al., 2007; Stepanyants et al., 2009; Ohki and Reid, 2007; Thomson and Lamy, 2007).

Furthermore, the cortex is not only an ensemble of anatomical, interconnected elements, but rather a system with complex dynamics. Describing the dynamical properties of the cortex was therefore the aim of many studies.

On the single neuron level, spike trains are irregular and show a large variability of spike counts (Softky and Koch, 1993; Shadlen and Newsome, 1998). The overall firing rates are low, in the order of few Hz (Brecht and Sakmann, 2002) and even values smaller than 0.1 Hz have been reported by Kerr et al. (2005). However, also high firing rates with regular spiking patterns during high amplitude stimuli have been reported. The impinging excitatory and inhibitory synaptic inputs are dynamically balanced (Haider et al., 2006) such that the average membrane potential stays a few millivolts below the firing threshold. Spikes are thus elicited by membrane potential fluctuations (Gerstein and Mandelbrot, 1964; Shadlen and Newsome, 1998). The massive number of excitatory and inhibitory stimulation sources induces the "high-conductance state" of cortical neurons (Destexhe et al., 2003). *In vivo* recordings have shown that this high-conductance state changes the integrative properties of the neurons (see Destexhe et al., 2003; Léger et al., 2005 and Sections 1.3.1, 4.1.2).

[1] Jens Kremkow is with the Institut de Neurosciences Cognitives de la Méditerranée, CNRS, Marseille, France, and with the Institute for Neurobiology and Biophysics, University of Freiburg, Germany.

Characterizing the cortical dynamics on the network level is a more challenging task. Generally, the correlation of the spiking activity of neurons in a population is low (Abeles, 1991). However, it was shown that the correlation of the spiking activity of neurons is a function of their spatial distance and of neuronal tuning properties (Smith and Kohn, 2008). Similarly, based on intra-cellular recordings from pairs of neighboring neurons, not only synchronous membrane potential fluctuations have been reported (Lampl et al., 1999), but also high correlations of excitation and inhibition in the recorded neurons (Okun and Lampl, 2008).

Therefore, depending on the spatial scale, the cortical spiking activity can be classified as follows: Individual neurons can show irregular (I) or regular (R) inter-spike intervals. On the population level, the neurons can fire asynchronously (A) or synchronously (S). Thus, the resulting four characteristic activity states of cortex are: AI, AR, SI and SR.

The *in vivo* type activity of a large population of neurons is best described by the AI activity characteristics. Statistical measures to characterize and discriminate these states have been proposed (Brunel, 2000; Kumar et al., 2008) and are introduced in Section 5.3.2.

Activity States of Generic Random Networks

Despite the complex nature of the cortical architecture, the random recurrent network model has emerged as a standard theoretical model for the local cortical volume. This model has been subject to many studies and has been shown to be a useful tool to understand the dynamical properties of cortical networks (van Vreeswijk and Sompolinsky, 1996; Amit and Brunel, 1997; Brunel, 2000; Vogels and Abbott, 2005; Kumar et al., 2008; Legenstein and Maass, 2007; Maass et al., 2004b; Sussillo et al., 2007; Shelley et al., 2002).

Usually, it is composed of only two populations of neurons, one excitatory and one inhibitory, which are modeled as leaky I&F neurons with current- or conductance-based synapses. The excitatory and inhibitory neurons are randomly connected with a connection probability of around 10-20%, creating a sparsely interconnected network. Input from the surrounding cortical network is mimicked by providing spikes from uncorrelated point-processes (usually Poisson-distributed spike times)

Despite this simplistic nature, the model shows complex dynamics. Depending on parameters like the external input rate and the strength of the recurrent inhibition, the network elicits different dynamical activity states, closely resembling those observed *in vivo* and described above (Brunel, 2000; Vogels and Abbott, 2005; Kumar et al., 2008). Furthermore, such networks can exhibit self-sustained activity (Vogels and Abbott, 2005; Kumar et al., 2008), which has been observed in isolated brain tissues (Plenz and Aertsen, 1996; Timofeev et al., 2000).

The capability of the random network to reproduce *in vivo*-like activity in terms of statistical measures renders it a suitable candidate for comparing the VLSI system with software simulations on a statistical level.

Implemented Cortical Network Model

The architecture used in the following is composed of an excitatory and an inhibitory population. The total neuron number is N_net, with the larger fraction of all neurons being excitatory, $N_\mathrm{e} = 0.75 \cdot N_\mathrm{net}$, and the remaining being inhibitory, $N_\mathrm{i} = 0.25 \cdot N_\mathrm{net}$. The neurons are sparsely connected with well defined intra- and inter-population connection probabilities, and they receive independent external Poisson-type input spike trains, thereby mimicking cortical background activity.

The probabilities $p_{\mathrm{xy}} \in \{p_{\mathrm{ee}}, p_{\mathrm{ei}}, p_{\mathrm{ie}}, p_{\mathrm{ii}}\}$ determine if a synapse from a neuron of type x to another neuron of type y is created, where e stands for *excitatory* and i for *inhibitory*. The parameters $g_{\mathrm{xy}} \in \{g_{\mathrm{ee}}, g_{\mathrm{ei}}, g_{\mathrm{ie}}, g_{\mathrm{ii}}\}$ denote the corresponding synaptic weights, i.e. the amplitudes of the quantal conductance increases as a response to spikes arriving at these synapses(see Section 2.1.2). In the following, the probabilities for connections *from* neurons of the same type get the same value: $p_{\mathrm{i}} \equiv p_{\mathrm{ii}} = p_{\mathrm{ie}}$ and $p_{\mathrm{e}} \equiv p_{\mathrm{ee}} = p_{\mathrm{ei}}$. The same accounts for the corresponding synaptic weights: $g_{\mathrm{i}} \equiv g_{\mathrm{ii}} = g_{\mathrm{ie}}$ and $g_{\mathrm{e}} \equiv g_{\mathrm{ee}} = g_{\mathrm{ei}}$.

N_{ext} is the number of externally generated Poisson processes used to activate the neurons, p_{ext} is the probability for spike sources to get connected to any neuron within the network. The corresponding synaptic weights of these purely excitatory stimulus synapses is g_{ext}, and every process fires with a spike rate of ν_{ext}.

Figure 6.10 shows a schematic diagram of the described network architecture. The complete set of applied parameter values is listed in Section 6.2.1. Some value choices are determined by constraints imposed by the hardware system – they will be motivated in the following paragraph.

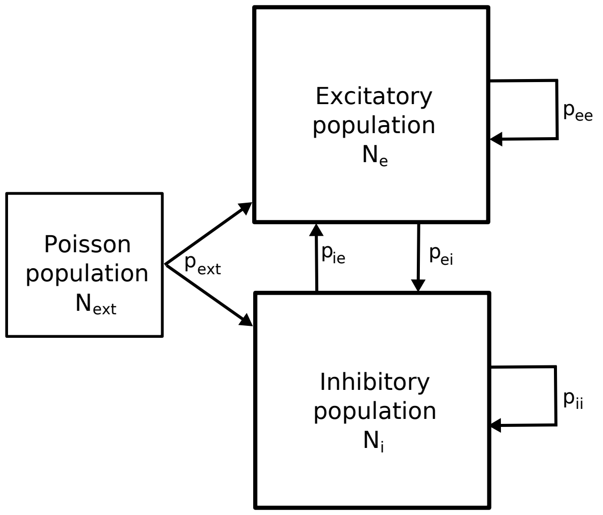


Figure 6.10: Schematic diagram of the network architecture. N_{e} and N_{i} are the numbers of excitatory and inhibitory neurons, respectively. The labels $p_{\mathrm{xy}} \in \{p_{\mathrm{ee}}, p_{\mathrm{ei}}, p_{\mathrm{ie}}, p_{\mathrm{ii}}\}$ for each arrow indicate the probability of making a synapse of a neuron of type x onto a neuron of type y, where e stands for *excitatory* and i for *inhibitory*. N_{ext} is the number of externally generated Poisson processes used to activate the neurons, p_{ext} is the probability for every possible stimulus-to-neuron connection to be established.

Realization on the Hardware Substrate

The networks studied in (Brunel, 2000; Kumar et al., 2008) consist of up to 10^5 neurons, which is a realistic size for approximately 1 mm^3 of mammalian cortex. Due to architectural constraints of the utilized neuromorphic hardware system, a network size in that range is not realizable (see Section 2.1): On an `FHW-1` chip, the largest set of neurons with a connectivity structure flexible enough for the kind of experiments presented here is a so-called *network* *block* (see Section 2.1.2). One network block implements 192 neurons and 256 synapse drivers which provide the spike input to the neurons. Each synapse driver receives its input either from an external playback memory or from one of the 192 neurons. It delivers its input signal to a row of 192 programmable synapse nodes, which can be configured to connect the synapse driver to any of the 192 neurons within the block via individual weights. This results in a synapse array of $256 \times 192 = 49152$ programmable synaptic weights with a four bit resolution each.

Thus, if configured adequately, the neurons on one network block of the chip are fully connectable among each other. But the number of spike sources for all 192 neurons in total is limited to a number of 256, including the feedback connections between these neurons. For the present experiments, 192 out of the 256 synapse drivers were reserved for the recurrent network connections, providing the wiring for all theoretically possible recurrent connections. The remaining 64 synapse drivers were used to feed in externally generated Poisson-type background activity, leading to the constraint that the background input to each neuron cannot consist of purely independent spike trains, but rather of samples from a pool of 64 Poisson sources, drawn with a certain probability p_{ext}. Thus, for randomly chosen pairs of neurons, p_{ext} defines the amount of common input.

As a further constraint, the `FHW-1` chip has an upper limit of spikes that can be delivered to each synapse driver per time interval (see Section 4.3.7). For the applied hardware speedup factor of 10^5, this limit is in the order of $\nu_{\mathrm{ext}} \approx 12\,\mathrm{Hz}$ (`BTD`, see Sections 4.3.7 and 6.1.2). Due to this low number of spikes per Poisson process, it is even more necessary for each neuron to sample from multiple Poisson processes in order to receive enough spikes to reach its firing threshold. Very strong synapses are to be avoided, otherwise an average membrane potential fluctuating closely below the firing threshold – a regime that is typical for AI activity – would not be possible. Consequently, in the chosen setup, the background input to the neurons is more correlated and not as independent as described in (Brunel, 2000; Kumar et al., 2008). However, considering that with such a small number of neurons the network represents only the very local neighborhood, correlations between membrane potentials are expected (see Section 6.2.1 and Lampl et al., 1999; Okun and Lampl, 2008).

Experimental Procedure

For one full experiment, all parameters except of ν_{ext} and g_{i} are kept constant. The two free parameters span a space which is sampled by sub-experiments in a two-dimensional grid. Every sub-experiment consists of n_{stat} repetitions of the same setup, but with new random choices for both the network connectivity and the external stimulation patterns. Such two-dimensional parameter space sweeps are performed both with the `FHW-1` hardware model and with the software simulator NEST. In an ideal setup, the spiking activity of the full network is recorded. The following statistical descriptors, introduced in Section 5.3.2, are applied to the spikes recorded from every single run:

- The mean population firing rate ν_{net} as a measure for global activity,
- The averaged pairwise (neuron to neuron) correlation coefficient $\mathrm{CC}_{\mathrm{Sync}}$ for the network synchrony.
- The averaged squared coefficient of variation CV^2 of the individual inter-spike interval distributions for the irregularity of firing.

For every sampled point in the parameter space, the average of ν_{net}, CC_{Sync} and CV^2 over all n_{stat} runs is determined. This results in three two-dimensional data point grids for one experiment.

Back-End Matching Methods

The following techniques are applied in order to optimize the matching between the network dynamics generated by the hardware system and the software simulator.

Hardware Tuning The following methods aim at the establishment of a homogeneous hardware substrate:

- For the hardware back-end, the membrane time constants are calibrated with the method described in Section 5.2.3.
- The hardware synapse driver efficacies and time constants are calibrated according to Section 5.2.4. In order to minimize distortions caused by the activity-dependent efficacy fluctuations of these drivers (see Section 4.3.4), this calibration is performed with an architectural setup and with activity regimes as close to this experiment as possible.
- The firing threshold vs. reset potential trade-off calibration described in Section 5.2.2 is applied to all hardware neurons. This results in the exclusion of some hardware neurons from the set of usable cells, because they exceed the defined tolerance limits. Consequently, there are even less than 192 neurons available for the realization of the experiment in hardware. In order to keep this experiment setup transferable to different chips, a value $N_{\text{net}} = 160$ is chosen, so that up to 32 neurons per network block can be rejected. The neurons that are actually used have to be mapped to cells on the chip that are connected to synapse drivers calibrated for the correct type, i.e. excitatory or inhibitory.
- The occurrence of multi-spikes as described in Section 4.3.6 is filtered out for the spike recording. In case of two successive output spikes from the same neuron with an inter-spike interval smaller than 0.3 ms (`BTD`), only the first spike is kept, the second is dropped. This can only be done offline with a dedicated software routine, i.e. the multi-spike problem is not solved for the feedback connection within the network.
- The experiment is described fully in PyNN (see Section 3.1.3). Setup, operation and output data analysis are controlled with the same set of scripts for both NEST and the hardware, all from within a single Python environment. At a few positions within the PyNN setup code, the sequential execution forks into two branches, each of which is executed by only one of the two back-ends. The branches always merge into one code sequence after a few instructions, but they are necessary to incorporate fluctuations into the NEST model that mimic the hardware imprecisions. This NEST extension will be explained below.

Still, due to the set of problematic hardware issues described in Section 4.3, the two modeling platforms cannot be expected to behave similarly at every level of activity after the application of these methods – not even in terms of statistical measures like the averaged network firing rate (see also Section 6.1.2).

NEST Model Modifications A perfect hardware substrate can never be achieved, and even after the optimization methods listed above have been applied, tolerance ranges accepted by the calibration algorithms remain as uncertainties in the behavior of the circuits. Hence, some of these remaining hardware-specific fluctuations are incorporated into the NEST model. This further improves the matching between the two systems and can possibly provide useful information about hardware-specific imprecisions which are not directly accessible. The following features, which shall mimic remaining hardware-specific imperfections, have been introduced into the NEST description of the model:

- The individual values of the firing thresholds are chosen from a normal distribution with its mean value being the model target value V_{thresh} and its standard deviation being $\sigma/\mu(V_{\mathrm{thresh}}) = 5\%$. This assumption is motivated by the following simple consideration: During the threshold-vs-reset calibration (see Section 5.2.2) of the `FHW-1.3-No.18` chip, for 17 out of the 384 neurons no configuration was found that puts the firing thresholds of these neurons into the tolerance range of $V_{\mathrm{thresh}}^{\mathrm{target}} \pm 10\%$. Generally, for random samples from a normal distribution, approximately 96% are lying within $\mu \pm 2 \cdot \sigma$. Assuming a normal distribution for the hardware thresholds caused by transistor-level variations, the fraction of $17/384 \approx 4\%$ rejections at a tolerance of 10% indicates a relative standard deviation for the threshold of 5%. This assumption will be checked in preparation studies presented further below.

- The individual values of the synaptic weights are chosen from a normal distribution, with a mean value of g_{e} respectively g_{i}, and with a standard deviation being 20% of this mean. The chosen distribution width is motivated by preparation studies described below.

- The weights of the excitatory feedback connections are multiplied with a factor f_{e}, which is used to fine-tune the matching of the global firing rates. The corresponding tuning experiment will also be described further below.

Recording from Subsets Due to the spike recording deadlock issue described in Section 4.3.1, the output spikes of only a subset of hardware neurons can be recorded at the same time. Therefore, the PyNN script considers this and, both when using the hardware back-end and when using NEST, acquires data only from those neurons which are known to be recordable in hardware. This limitation is transferred to NEST in order to keep the results comparable. Otherwise, statistical measures like the pairwise firing correlation might become distorted by sampling from a much larger pool of spike trains in the NEST case.

The sub-set recording raises the necessity for more runs per experiment in order to acquire enough statistics for spike-based analyses.

Parameter Values

Due to the mentioned resource constraints and technical obstacles imposed by the hardware system (see Sections 6.2.1, 2.1 and 4.3), the parameter values for the network connectivity are chosen different from those used in (Brunel, 2000; Kumar et al., 2008).

In particular, the parameter values used for the main experiment are listed in Table 6.1. Unless otherwise expressly mentioned, the preparation studies described below use the same values.

Description	Parameter	Unit	Value
Network Architecture			
Number of excitatory neurons	N_e		120
Number of inhibitory neurons	N_i		40
Connect prob. from exc. to exc. neurons	p_ee		0.25
Connect prob. from exc. to inh. neurons	p_ei		0.25
Connect prob. from inh. to exc. neurons	p_ie		0.5
Connect prob. from inh. to inh. neurons	p_ii		0.5
Neurons (Excitatory and Inhibitory)			
Reset potential	V_reset	mV	-80.0
Inhibitory reversal potential	E_i	mV	-80.0
Leakage reversal potential	E_l	mV	-75.0
Firing threshold voltage	V_thresh	mV	-55.0
Excitatory reversal potential	E_e	mV	0.0
Leakage conductance	g_l	nS	40.0
Synapses			
Cond. amplitude for excitatory internal synapses*	g_e	nS	1.88
Cond. amplitude for inhibitory internal synapses	g_i	nS	(swept) 3.0 .. 15.0
Cond. amplitude for synapses from external sources*	g_ext	nS	2.82
Cond. time constant for all synapses	τ_syn	ms	30.0
External Stimulus: Poisson Spike Trains			
Number of external Poisson spike sources	N_ext		64
Connect prob. from any ext. source to any neuron	p_ext		0.25
Firing rate per spike train	ν_ext	Hz	(swept) 4 .. 10
Experiment			
Simulated / Emulated time per experiment run	T_exp	ms	10,000
Number of experiment runs per parameter set	n_stat		200

Table 6.1: Full set of experiment parameters. All values given in BVD and BTD. For the parameters marked with an asterisk (*), the given values are those written to the hardware system. For the software runs, the values are multiplied by scaling factors in order to simulate hardware-specific efficacy fluctuations (see the following section *Preparation Studies*).

Preparation Studies

Weight Distribution The synapse driver calibration routine described in Section 5.2.4 homogenizes the impact that the different synapse drivers have on a neuron membrane. By utilizing PyNN, the calibration algorithm also performs a matching with the reference simulator NEST. Still, after the calculated calibration values are applied, a permanent imprecision of the synapse driver efficacies remains: The standard deviation of the PSP integrals caused by the individual synapse drivers is approximately 15% of the mean value over these integrals. This fluctuation estimation accounts for single neurons and was determined in a setup with well defined membrane dynamics. The estimator does not incorporate possible fluctuations caused by different synapse node efficacies or any activity dependencies – which have to be expected due to a design-related malfunction of the `FHW-1` system described in Section 4.3.4.

In a preparing NEST experiment, where such weight fluctuations can be added, the influence of the width of a Gaussian input weight distribution on the firing behavior of the considered neurons is analyzed. For this purpose, the network architecture defined above is exposed to 64 Poisson spike trains with $\nu_{\mathrm{ext}} = 10\,\mathrm{Hz}$ (`BTD`) each, while its recurrent connections are deactivated. Hence, every neuron receives a subset of these 64 spike trains, but no feedback from other neurons. The resulting mean population firing rates ν_{net} is acquired in 30 runs of $T_{\mathrm{exp}} = 10\,\mathrm{s}$ (`BTD`) length each. In every run, new Poisson patterns and new connections between the stimuli and the neurons are randomly generated. The mean value $\mu\left(\nu_{\mathrm{net}}\right)$ and the standard deviation $\sigma\left(\nu_{\mathrm{net}}\right)$ of ν_{net} over all 30 runs are computed.

In multiple repetitions of this experiment series, the width of the underlying Gaussian weight distribution $\sigma_{\mathrm{g}}^{\mathrm{NEST}}$ is varied, and the NEST values for $\mu\left(\nu_{\mathrm{net}}\right)$ as a function of this width are determined. Figure 6.11 shows the results: The data points with error-bars represent the NEST measurements of $\mu\left(\nu_{\mathrm{net}}\right)$ and $\sigma\left(\nu_{\mathrm{net}}\right)$ for different values of the weight distribution width $\sigma_{\mathrm{g}}^{\mathrm{NEST}}$. The dashed-dotted line indicates a constant fit, which, considering the standard deviations of each data point, suggests that for the applied widths the influence of the weight distribution can be ignored. This results from a sufficiently effective averaging performed by every neuron membrane over its multiple input synapses.

The measures $\mathrm{CC}_{\mathrm{Sync}}$ for the population synchrony and CV^2 for the single neuron firing regularity were found to be not significantly influenced by $\sigma_{\mathrm{g}}^{\mathrm{NEST}}$, either. Therefore, for all following experiments in this section, a weight distribution width of $\sigma_{\mathrm{g}}^{\mathrm{NEST}} = 0.2 \cdot g$ is assumed. This value represents the fluctuations that remain after the hardware synapse driver calibration, and additionally incorporates a careful estimation of further fluctuations imposed by the hardware synapse nodes.

Stimulus Weight Tuning Due to a design-related malfunction that results in activity-dependent efficacies of excitatory hardware synapses (see Section 4.3.4)and due to parasitic capacitances in the synaptic signalling path of the `FHW-1` system (see Section 6.1.2), the weights of all stimulation synapses in NEST are tuned such that the average output firing rate in a network without internal feedback fits the hardware results.

The basic setup as listed in Table 6.1 is applied, but again p_{ee}, p_{ei}, p_{ie} and p_{ii} are set to zero. Hence, every neuron receives a subset of the 64 external Poisson-type spike trains, but no feedback from other neurons. The stimulation spike trains fire with a frequency of $10\,\mathrm{Hz}$ (`BTD`) each, and in 30 hardware runs the resulting population firing rate ν_{net} (see Section 5.3) is determined. This results in a mean value $\mu(\nu_{\mathrm{net}}) = 7.8\,\mathrm{Hz}$ and a standard deviation $\sigma(\nu_{\mathrm{net}}) = 2.4\,\mathrm{Hz}$ (`BTD`) over these 30 runs. The same experiment series is performed

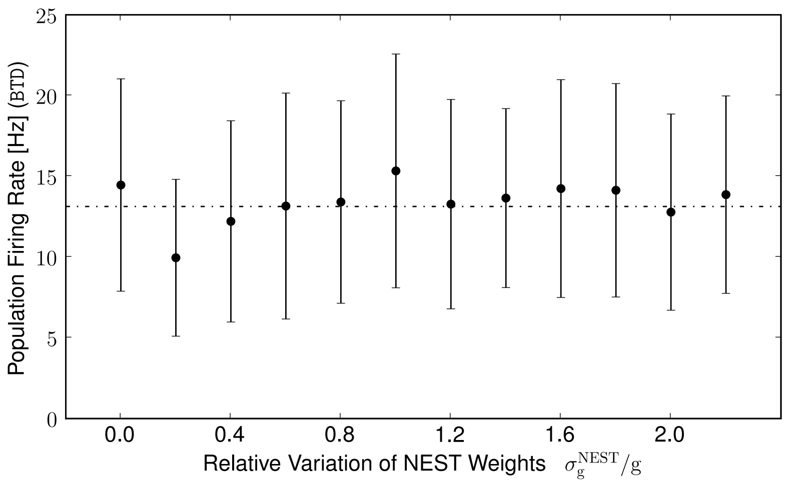


Figure 6.11: NEST simulations: 160 unconnected neurons receive Poisson-type input stimuli, sampling from a set of 64 spike sources with a connection probability of $p_{\mathrm{stim}} = 0.25$. The width of the distribution from which the synaptic weights for these stimuli are drawn is varied. Every data point represents the mean population firing rate $\mu(\nu_{\mathrm{net}})$ over 30 runs of the same experiment, the error-bars are the corresponding standard deviations. The dashed-dotted horizontal line indicates a constant fit through the data (red. $\chi^2 = 2.4$).

multiple times with the simulator NEST, each time with another factor f_{stim} multiplied to the original weights g_{ext}. Figure 6.12 shows the results: A good matching between the NEST data (data points with error-bars) and the hardware result (full and dashed lines) is achieved for $f_{\mathrm{stim}} \approx 0.63$.

It has to be assumed that this factor depends on the applied threshold fluctuations in NEST, which change the diversity of responsiveness for the network neurons. Hence, in addition to the mean population firing rates, further criteria of dynamics similarity between the `FHW-1` system and NEST have to be applied.

Regularity and Synchrony of Firing The degree of firing synchrony within the network is assumed to depend on the diversity of the involved firing thresholds. If two neurons receive very similar input, but have different firing thresholds, their response will not be as synchronous as in the case of two identical neurons. The same accounts for the regularity of firing of individual neurons. Two scenarios can strongly decrease the width of inter-spike intervals generated by a neuron: If the neuron is stimulated very strongly via excitatory synapses, its firing behavior can be mainly determined by its refractory period, i.e. it will fire as fast as it can, with very regular and short periods between the spikes. Another possibility is the emergence of a global activity oscillation. The probability of such a population effect decreases with a higher diversity of responsiveness among the neurons.

Hence, changing the width of the firing threshold distribution is assumed to have an impact on both the population firing synchrony as well as on the regularity of firing for every single neuron. The following setup tests if this impact is significant: While the threshold distribution width in NEST is varied, the measures $\mathrm{CC}_{\mathrm{Sync}}$ for network synchrony and CV^2 for the network firing irregularity are acquired and compared with the corresponding hardware results.

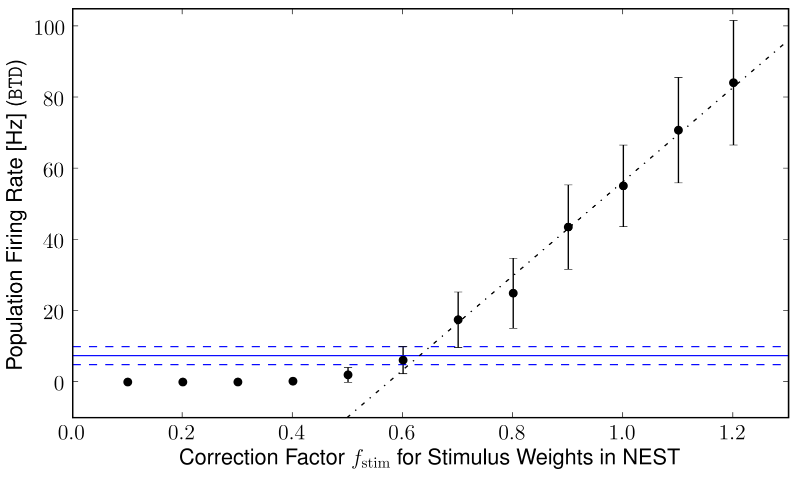


Figure 6.12: Hardware and NEST: Firing rate matching by sweeping a tuning factor for the stimulation weights in the NEST simulations. A set of hardware neurons receives Poisson-type input stimuli via synapses which are all configured with the same weight value. But due to hardware-specific issues, the effective weight values decrease for higher loads on the excitatory reversal potential. The full horizontal line represents the mean population firing rate $\mu(\nu_{\mathrm{net}})$ over 30 hardware runs of the same experiment. The dashed horizontal lines indicate the standard deviation $\sigma(\nu_{\mathrm{net}})$ over these 30 hardware runs. With the software simulator NEST, the same experiment is performed, but with a varying tuning factor f_{stim} for the weights of the excitatory stimulation synapses (horizontal axis). The data points with the error-bars represent the corresponding mean values $\mu(\nu_{\mathrm{net}})$ and the standard deviations $\sigma(\nu_{\mathrm{net}})$ over 30 NEST runs. The value of f_{stim} is searched which fits the hardware results best. A linear fit through the NEST data with $\nu_{\mathrm{net}} \leq 5\,\mathrm{Hz}$ (BTD) suggests an optimal factor of $f_{\mathrm{stim}}^{\mathrm{opt}} \approx 0.63$.

For different values of $\sigma(V_{\mathrm{thresh}})$, the input weight correction factor f_{stim} that matches the NEST and the hardware firing rates best is determined as described above ("Stimulus Weight Tuning"). For this optimal correction factor, the values of $\mathrm{CC}_{\mathrm{Sync}}$ and CV^2 are determined for both NEST and the hardware. Table 6.2 summarizes the results. For every applied threshold distribution width, a factor f_{stim} can be found that results in matching population firing rates of the hardware system and NEST. As expected, this value varies with the applied $\sigma(V_{\mathrm{thresh}})$. For all NEST runs, the synchrony measure $\mathrm{CC}_{\mathrm{Sync}}$ is smaller than the corresponding hardware value. The values of the irregularity measure for the NEST runs are in the same region as the corresponding hardware value, but do not exactly match either. Both $\mathrm{CC}_{\mathrm{Sync}}$ and CV^2 are not significantly affected by the choice of $\sigma(V_{\mathrm{thresh}})$.

It has to be concluded that the hardware seems to generate more synchronous firing behavior by means of the applied measure. One possible reason for this is the damping effect caused by parasitic capacitances in the hardware transmission path of synaptic signals. This might cause an intrinsically increased responsiveness of individual hardware neurons to temporally dense input signals. Such a behavior would result in a selectively amplifying effect for synchronous input sequences. Experimental observations presented further below will support this assumption.

In this pure stimulation setup without feedback, the chosen synchrony and firing regularity measures do not provide additional information about the correctness of the assumed threshold fluctuations. Therefore, the threshold distribution width is kept at the previously

	Threshold Distribution Width $\sigma/\mu(V_\mathrm{thresh})$	Weight Factor f_stim	Synchrony $\mathrm{CC}_\mathrm{Sync}$	Irregularity CV^2
Hardware Emulation				
	?	1	0.15	1.6
NEST Simulation				
	0%	0.66	0.09	2.2
	2%	0.65	0.08	2.3
	4%	0.64	0.09	2.2
	6%	0.60	0.09	2.0
Error Estimation NEST values		± 0.05	± 0.01	± 0.3

Table 6.2: A set of hardware neurons is stimulated with Poisson-type spike trains. The measures $\mathrm{CC}_\mathrm{Sync}$ for the resulting firing synchrony among the stimulated neurons, and CV^2 for the irregularity of the firing of individual neurons, are listed. In NEST simulations of the same experiment, the firing thresholds of the stimulated neurons are drawn from a distribution with varying widths $\sigma(V_\mathrm{thresh})$. For every width, a weight correction factor f_stim is determined which results in population firing rates that match the hardware result. For these matching setups, the values of $\mathrm{CC}_\mathrm{Sync}$ and CV^2 are listed and compared with the corresponding hardware results. Only actively spiking neurons were considered for the calculation of $\mathrm{CC}_\mathrm{Sync}$ and CV^2. See main text for an interpretation of the data.

motivated value of $\sigma(V_\mathrm{thresh}) = 0.05 \cdot V_\mathrm{thresh}$ for the following experiments.

Excitatory Feedback Strength The frequently mentioned circumstance that the strength of excitatory `FHW-1` synapses varies with the load on the excitatory reversal potential (see Section 4.3.4) causes a serious problem for the experiment described in this section. A wide range of different network firing rates is expected to be generated with the planned parameter scans. This results in many different and a priori unpredictable levels of load on the excitatory reversal potential. Therefore, the described hardware malfunction cannot be systematically counterbalanced with individual weight calibration factors for every activity state. In the following NEST experiments, all excitatory feedback weights are multiplied with a factor of $f_\mathrm{e} = 0.5$, independent of the level of network activity or external stimulation. As will be shown in the following, this value results in a good network firing rate matching between hardware and software for a wide range of the tested parameter sets.

Results and Interpretation

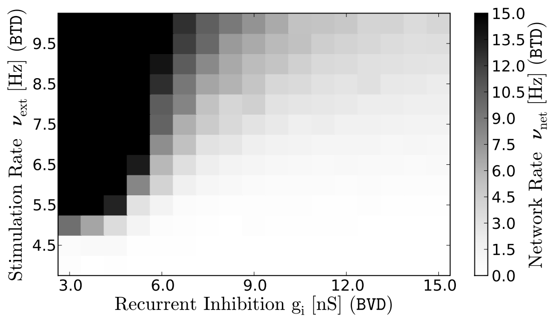
After these preparation studies, the main experiment as described above ("Experimental Procedure") is executed both on the `FHW-1` system and with the software simulator NEST. The two parameters ν_ext and g_i are scanned. For every considered point in the two-dimensional parameter space, $n_\mathrm{stat} = 200$ experiment runs with a simulated period of 10000 ms (`BTD`) are performed. The externally applied stimulus patterns and the synaptic connections are randomly re-generated for each of these runs.

Activity States of the Emulated and Simulated Model In Figure 6.13, the values of the firing rate ν_net, the synchrony measure $\mathrm{CC}_\mathrm{Sync}$ and the irregularity descriptor CV^2 are plotted for all scanned parameter sets. Every data point represents the average over 200 runs. The hardware results are drawn in the left column (sub-figures **a**, **c** and **e**), the NEST results in the right (sub-figures **b**, **d** and **f**).

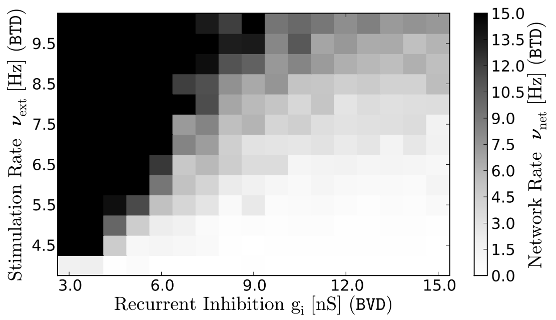
The average network firing rates emulated with the hardware system and computed with NEST match well. Although tuning factors had to be applied to the excitatory NEST weights in order to simulate hardware-specific problems, the reproduction of the diagonal line of activity onset is remarkable.

The irregularity measures acquired from both back-ends exhibit a coarse qualitative matching. In both cases, the values of CV^2 are strongly correlated with the network firing rate ν_{net}, i.e. the regions of high firing irregularity are approximately the same as the regions of high activity. For the hardware measurements, though, the upper left corner in the CV^2 plot indicates an increased regularity of firing. Such a region cannot be observed in the corresponding NEST plot. A possible explanation of this behavior is the increased load on the excitatory reversal potential due to the strong network activity. The resulting loss of excitatory synaptic efficacy in the hardware system (see Section 4.3.4)can possibly cause oscillation phenomena. In such a scenario, activity peaks temporarily weaken the excitatory reversal potential. This weakening results in a decrease of network activity. Consequently, the excitatory reversal potential can recover, and the network activity increases again.

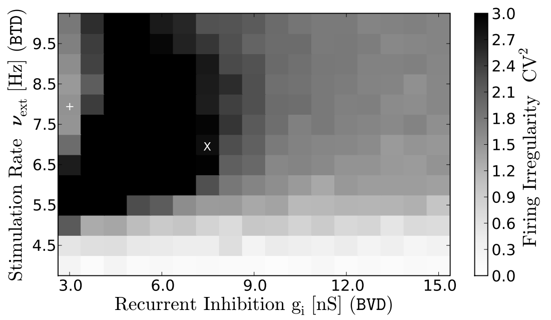
The synchrony measure CC_{Sync} reveals another interesting discrepancy between the hardware system and NEST: While the network-internal inhibition determines the firing synchrony on both back-ends, the external stimulation rate affects the hardware system much stronger than the NEST networks. For low stimulation rates around 5.5 Hz (`BTD`), the hardware neurons exhibit systematically increased values of CC_{Sync}. This can be explained by parasitic capacitances in the `FHW-1` transmission path of synaptic signals, which result in decreased synaptic efficacies for low input rates per synapse (see Section 6.1.2). Such a damping can cause a network-wide, selectively amplifying effect for temporally dense input sequences.



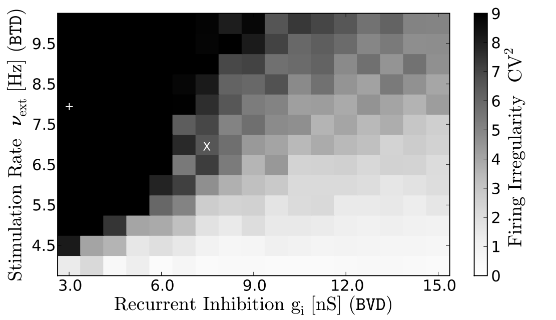
(a) Hardware: Firing Rate ν_{net}



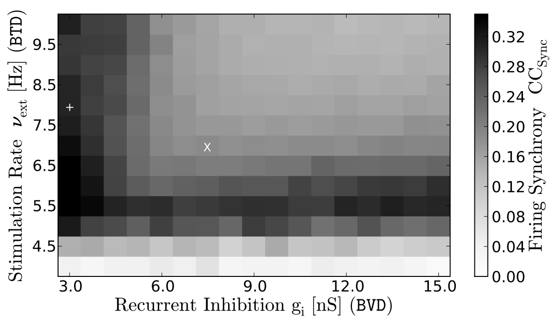
(b) NEST: Firing Rate ν_{net}



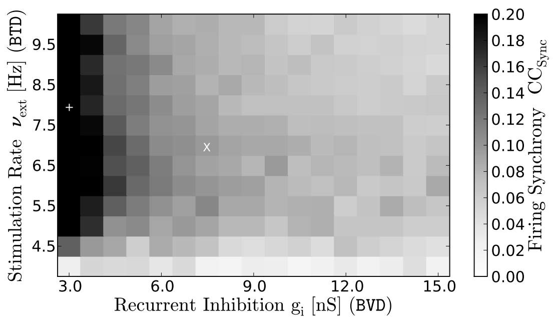
(c) Hardware: Irregularity CV^2



(d) NEST: Irregularity CV^2



(e) Hardware: Synchrony $\mathrm{CC}_{\mathrm{Sync}}$



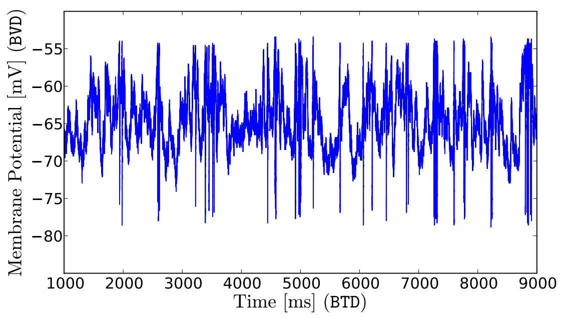
(f) NEST: Synchrony $\mathrm{CC}_{\mathrm{Sync}}$

Figure 6.13: Statistical activity measures for a recurrent network of excitatory and inhibitory neuron as a function of the externally applied stimulus frequency ν_{ext} and the strength g_{i} of the inhibitory feedback synapses. The left column shows results acquired with the `FHW-1` system, the right column represents NEST simulations. Sub-figures **(a)** and **(b)** plot the average output firing rates ν_{net}, sub-figures **(c)** and **(d)** plot the irregularity measure CV^2, sub-figures **(e)** and **(f)** plot the synchrony measure $\mathrm{CC}_{\mathrm{Sync}}$. The white cross and plus symbols drawn in the lower four sub-figures indicate the parameters for which the network activity is analyzed in more detail (see following section). The upper limit of the color code does *not* necessarily represent the maximum value within a diagram, i.e. the color black denotes a value equal to that limit *or larger*. For every presented NEST data point, the determined standard error of mean is 10% of the plotted mean value or less. For the hardware data points, the same upper limit is true for all data points with an output firing rate of $\nu_{\mathrm{net}} \leq 1\,\mathrm{Hz}$ (`BTD`). For lower output rates, erroneous ghost events generated by the chip (see Section 4.3.12) result in relative standard errors of mean of up to 40%.

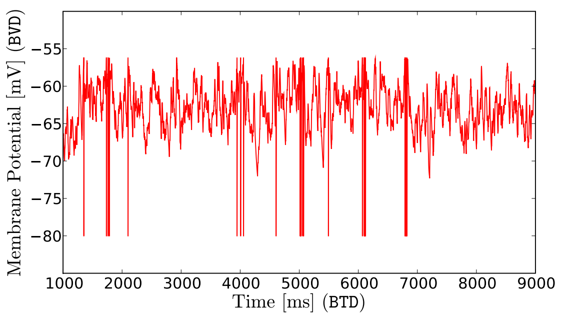
Detailed Comparison of Two Activity States

To further illustrate the different network dynamics occurring in two distinct activity states: One candidate for an AI state and one for an instable SI/SR state. Figure 6.14 and Figure 6.15 show the membrane potential trace of one neuron and the raster plots of several recorded active neurons.

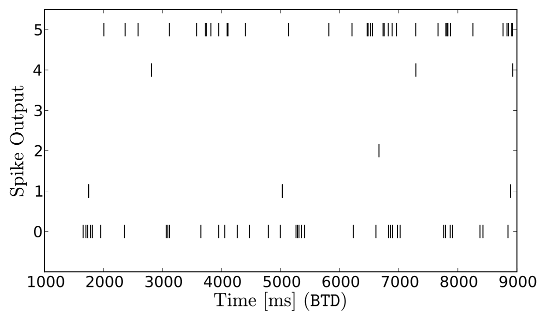
The AI activity state is shown in Figure 6.14. The parameter combination that has been used to generate this state is marked by a white cross in Figure 6.13. The membrane potential of both back-ends fluctuates around a mean value a few millivolts below the firing threshold (**a** and **b**). The spikes are elicited by these fluctuations, inducing irregular spiking pattern. The correlation across the neurons is low, as can be seen in the raster plots of 5 neurons (**c** and **d**). Both the sub- and the supra-threshold activity of both back-ends is comparable during this activity state.



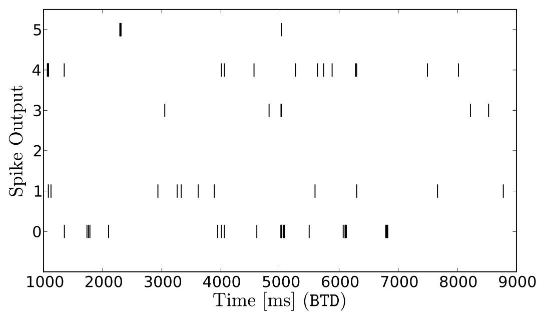
(a) Hardware: Membrane Potential



(b) NEST: Membrane Potential



(c) Hardware: Spike Trains



(d) NEST: Spike Trains

Figure 6.14: Characteristic membrane potential trajectories and spiking responses as measured during an AI activity state. The left column shows results acquired with the `FHW-1` system, the right column represents NEST simulations. Sub-figures **(a)** and **(b)** plot the membrane potential trace of one recorded neuron, sub-figures **(c)** and **(d)** plot the raster plot of the spiking behavior of 5 neurons.

In contrast to the stable AI state, the membrane potential and spike output samples shown in Figure 6.15 illustrate a very instable network situation. The parameter sets for the sampled examples are marked by a white plus in Figure 6.13. For a certain initial period, which is determined by the randomly applied input patterns, the network exhibits no output activity at all. But as soon as a few neurons start to fire, the excitatory feedback within the network leads to a strong global amplification of the activity. The figure shows such a situation for both back-ends, i.e. the illustrated time windows (note that the plotted periods are much smaller than in Figure 6.14) have been manually set to the onset of strong activity. From this moment of activity onset, most neurons fire with a maximum frequency determined by their refractory time and their effective membrane time constant (see Section 4.1.2), while others do not fire at all. This necessarily results in high values for the synchrony measure.

If all neurons fired at the same maximum rate permanently, the irregularity measure would be very small. But since the effective membrane time constant varies with the stimulation, and since the activity periods are sometimes interrupted (possibly caused by fluctuations in the input or by a temporarily dominating inhibition), the inter-spike interval histogram of such a firing pattern can be rather stretched. Furthermore, the squared average inter-spike interval, i.e. the denominator of the irregularity measure, is very small. Therefore, the total value of the irregularity measure is large.

The minimum inter-spike intervals in hardware are much larger than the corresponding NEST values. One possible reason for this is the way multiple spikes from one pre-synaptic source are integrated in the post-synaptic neuron. In NEST every spike from one source elicits the same conductance transient, which is added to the total conductance impinging on the neuron. In the hardware system, however, the situation is slightly different (see Section 2.1.2). A spike that arrives at the synapse driver induces a conductance transient, like in NEST. But with each new arriving spike this conductance transient is reset and re-starts from its beginning, independent of the preceding input history. Hence, in case of input rates arriving at a synapse driver that are larger than $1/\tau_{\mathrm{syn}}$, a considerable fraction of the stimulus impact gets lost, resulting in a nonlinear input-output relationship.

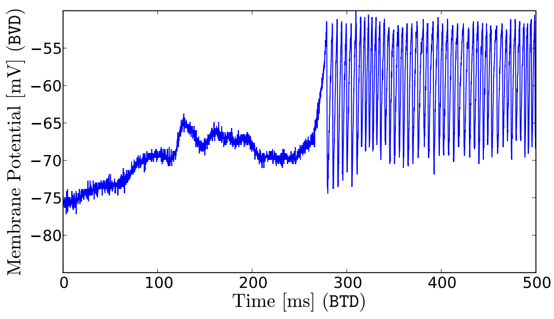
A further possible reason for the large hardware inter-spike intervals observed in Figure 6.15 are the instable excitatory reversal potentials (see Section 5.2.4). As described above, at high levels of global activity the excitatory reversal potential drops, which results in a reduced driving force and in reduced effective synaptic weights, respectively.

Runtime Comparison

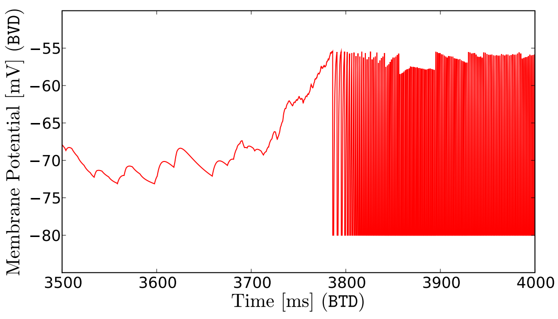
The data depicted in Figure 6.13 represents the statistical essence of a large number of single simulation runs. The 13×17 tested parameter sets with 200 runs of 10 sec (`BTD`) length each correspond to a total simulated time of $T_{\mathrm{bio}} = 13 \cdot 17 \cdot 200 \cdot 10\,\mathrm{s} = 442000\,\mathrm{s}$ (`BTD`). This corresponds to more than five days.

With the applied time translation factor of $f_{\mathrm{acc}} = 10^5$ between `HTD` and `BTD`, the total emulation time needed by the circuits of the `FHW-1` chip in order to generate the corresponding data is $T_{\text{on-chip}} = T_{\mathrm{bio}}/f_{\mathrm{acc}} = 4.42\,\mathrm{s}$ (`HTD`). The actual lab time that was consumed by the full system, i.e. with the overhead introduced by further components and with the calculation time for the statistical analysis included, was 72 hours. The main contributor to this significant slow-down is the inefficient network interface between the controlling host PC and the `FHW-1` carrier board (see Section 2.1.5), via which all hardware configuration and stimulation data as well as the generated output data has to be transported.

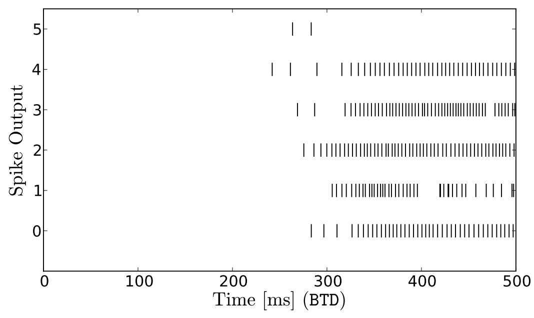
The NEST simulations have been performed on an Intel quad-core architecture, i.e. four simulations were executed at the same time on four processor cores in parallel, each of which is clocked with 2.4 GHz. The working memory of this computer shared by the four processor cores is 4 GByte. The total time consumed by this machine for the NEST simulations and the corresponding statistical analysis is 135 hours. It can be assumed that this quad-core



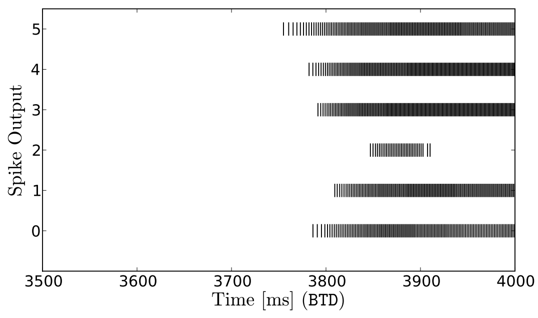
(a) Hardware: Membrane Potential



(b) NEST: Membrane Potential



(c) Hardware: Spike Trains



(d) NEST: Spike Trains

Figure 6.15: Membrane potential trajectories and spiking response as measured during an instable SI/SR activity state. Same arrangement as in Figure 6.14. For explanations see main text.

computer did not spend more time on the analysis of the data than the hardware host PC, because the analysis is not memory limited, and the hardware host PC is a single-core Intel architecture clocked with 2.4 GHz, too. Hence, even if 72 hours are assumed for pure data analysis, a total NEST execution time of 63 hours remains. This corresponds to 226800 seconds, i.e. once configured, the hardware generated the acquired data approximately 50000 times faster than the NEST software simulation. This comparison is not totally fair, since the considered NEST execution time also contains software-specific periods for data structure setup. Note that such an execution time comparison is very problem-specific. For larger numbers of neurons or synapses or for more firing activity in the network, the NEST simulation will require more execution time. This does not hold for the on-chip emulation time of the hardware system, which is only determined by the biological period to be emulated and by the hardware speedup factor f_{acc}.

Conclusions Drawn From This Section

The network experiments described in this section prove that the hardware-to-biology translation paradigms, the hardware tuning techniques and the comparison framework presented in this thesis provide an appropriate tool set to implement biologically motivated models with a neuromorphic device. The achieved level of similarity between the hardware system and the software simulator NEST is remarkable, especially if the imperfections of the device are considered.

For the remaining differences plausible explanations have been given, the most of which are design-related, i.e. they can be eliminated or minimized in future revisions of the hardware design. The following suggestions address the most critical issues. Significant improvements in the controllable generation of biologically realistic activity regimes with `FHW-1` devices are expected if these requirements will be satisfied:

- Reliable reversal potentials, that are not activity dependent.
- Reliable and independent configurability of the firing threshold and the reset potential.
- Full access to the spike output of all neuron circuits at the same time.
- An immediate synaptic response that is not initially damped by parasitic capacitances.

One further need for improvement has become obvious from the presented execution time comparison between the hardware emulation and the NEST simulation: A significantly faster interface between the controlling host PC and the hardware device is essential for the efficient exploitation of the speedup benefit intrinsic to the neuromorphic approach. A Gigabit Ethernet interface that will solve exactly this issue is currently under development in the Kirchhoff Institute in Heidelberg.

6.2.2 Self-Stabilizing Network Architectures

In Section 6.2.1, the realization of a recurrent network on an `FHW-1` device has been presented. That network consisted of excitatory and inhibitory neurons, and by varying the strength of the inhibitory internal feedback and the frequency of the applied stimuli, the network has been purposely put into different states of spiking activity, characterized by the frequency, the regularity and the synchrony of firing. In contrast to this, the goal of the following experiment series is not the generation of a large variety of dynamics, but the opposite: A hardware network shall be set up that exhibits a stable firing activity despite variations in the stimulation strength and in the responsiveness of the utilized neurons.

Such a working point stability is achieved by appropriately applying the short-term plasticity mechanisms implemented in the `FHW-1` synapses (see Sections 1.3.3 and 2.1.3). Before the experiment is realized on the hardware substrate, a preparation study based on pure software simulations proves the functionality of the chosen approach for the limited number of available hardware neurons.

The results presented in this section have been acquired by Johannes Bill under the supervision of the author (see also Bill, 2008).

Setup and Stabilization Mechanism

Inspired by the descriptions given in Sussillo et al., 2007, a network architecture consisting of two populations is set up: An excitatory population P_e, consisting of $N_\mathrm{e} = 144$ neurons, and an inhibitory population P_i, consisting of $N_\mathrm{i} = 48$ neurons. They are interconnected according to a specific pattern, which determines the depressing and facilitating type of the involved synapses:

- All synaptic connections from excitatory to excitatory neurons are depressing.
- All synaptic connections from excitatory to inhibitory neurons are facilitating.
- All synaptic connections from inhibitory to inhibitory neurons are depressing.
- All synaptic connections from inhibitory to excitatory neurons are facilitating.

All neurons within the network receive externally generated Poisson-type spike trains via static excitatory and inhibitory synapses. Figure 6.16 shows a schematic of the described connectivity scheme.

This architecture has an intrinsic self-stabilizing mechanism, which has been theoretically investigated and proved by the authors of Sussillo et al., 2007, and which becomes plausible by the following "differential" consideration: Given a state where the external stimulation has been constant for a while, and the population firing rates ν_e and ν_i are stable. If at this point the external stimulation increases, the firing rates of both populations will start to grow. As a consequence of this first response, the efficacy of the facilitating synapses increases, i.e. the excitatory feedback from P_e to P_i and the inhibition from P_i to P_e get stronger. Furthermore, all depressing synapses lose some of their efficacy, i.e. the intra-population feedback connections get weaker. Consequently, the activity in the excitatory population P_e is damped by two factors: By a weakening of its self-excitation and by an increased inhibition from P_i. Correspondingly, the activity in the inhibitory population P_i is amplified by two factors: By a weakening of its self-inhibition and by an increased excitation from P_e.

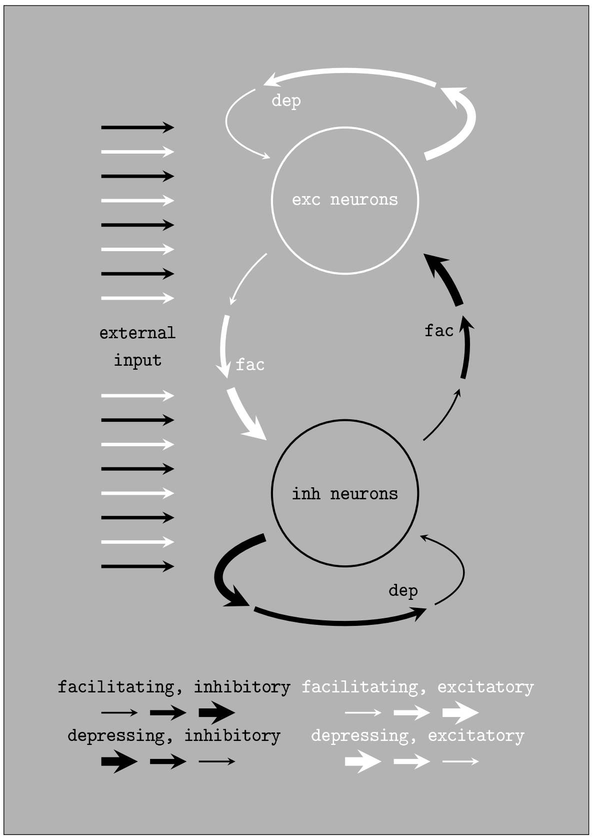


Figure 6.16: Schematic of a self-stabilizing network architecture inspired by Sussillo et al., 2007. Two populations of neurons, an excitatory (white circle) and an inhibitory one (black circle), receive externally generated input via static synapses. All network-internal synapses are dynamic: The intra-population connections are depressing, the inter-population connections are facilitating. The possible activity-balancing features arising from this are explained in the main text.

In the case of a weakened external stimulation, an analog consideration reveals that the activity in the excitatory population will be amplified, while the activity in the inhibitory population is damped.

The authors of Sussillo et al., 2007 show theoretically and with software simulations, that if the parameters of all synaptic connections within such an architecture are appropriately chosen, the network exhibits a low total firing rate ν_{net} that is very stable against changes in the external stimulation frequency.

Software Simulation Study

The simulated networks presented in Sussillo et al., 2007 consist of 5000 neurons, though, and all synapses exhibit a both facilitating and depressing behavior (Tsodyks and Markram, 1997). For the proposed network architecture, the maximum number of neurons that can be realized is 192, and every synapse can only be facilitating, depressing or static. Hence, Johannes Bill and Klaus Schuch tested the dynamics of such small and simplified architectures with the PCSIM software simulator (see Section 3.1.4) in order to prove that the same self-stabilization features occur. The software simulator allows to tune experiment parameters with an arbitrary flexibility. In contrast to the `FHW-1.3` system, which is subject to a readout deadlock issue (see Section 4.3.1), all neurons can be recorded at the same time, which is important for the determination of population firing rates. Furthermore, no possibly unknown malfunctions of the hardware short-term plasticity mechanisms distort the results, i.e. if the preparation studies with the software simulator do not prove that this kind of architecture can work with such few neurons, no futile effort needs to be invested into the search for phantom hardware bugs.

Applying Noise to the Parameters Similar to the experiment series presented in Section 6.2.1, certain parameters are drawn from a normal distribution D in order to simulate the variations in the hardware parameters. If the standard deviation σ_{D} of such a distributions is in the same order as the mean value μ_{D}, a policy has to be introduced that handles the cases in which the drawn value has a sign other than the mean value, but the parameter type does not allow this. Always setting such values to zero would result in a distortion of the resulting distribution. In order to avoid this, a new value is drawn from a uniform distribution in the range $[\mu_{\mathrm{D}} - s_{\mathrm{D}}, \mu_{\mathrm{D}} + s_{\mathrm{D}}]$ instead, where s_{D} is an adjustable so-called *spread*, with $|s_{\mathrm{D}}| < |\mu_{\mathrm{D}}|$.

Achieving an Uncorrelated Activation Level In order to keep the activity correlation within the network low, especially in order to avoid the emergence of strong, self-amplifying synchronization patterns, each neuron receives input only from a small subset of the applied 48 externally generated spike trains. Instead of applying very strong synaptic weights to reach the firing threshold with such few stimuli, a basis quasi-activation of the neurons is achieved by setting the leakage reversal potential V_{rest} to a value close to the firing threshold. Furthermore, in order to simulate the hardware-specific dynamic analog noise on the membranes (see Section 4.2.2), every neuron receives a fluctuating input current I_{noise} that is distributed normally around zero with a standard deviation of $\sigma_{\mathrm{I}} = 5\,\mathrm{pA}$ (`BVD`).

Test Procedure In order to test the firing rate stability of a given network architecture, two parameters are swept: The resting potential V_{rest} of all neurons, and a stimulation strength factor f_{stim}. The basis weights of all synapses that connect the externally generated spike trains with the network are multiplied by f_{stim}. Hence, this factor directly controls the efficacy of the externally applied excitation. The resting potential controls the responsiveness of the neurons, i.e. the closer this value is set to the firing threshold, the less synaptic stimulation is needed to make the cell fire.

The resulting two-dimensional parameter scans are performed for four versions of the network architecture:

- Case A: Weak internal connectivity, all synapses set to static.
  The basis values of internal connection weights are multiplied with a factor of $f_{\mathrm{int}} = 0.5$, and no depression or facilitation is incorporated in the network.

- Case B: Weak internal connectivity ($f_{\mathrm{int}} = 0.5$), synapses are set to be depressing and facilitating according to the architecture described in Figure 6.16.

- Case C: Strong internal connectivity ($f_{\mathrm{int}} = 3.0$), all synapses are static.

- Case D: Strong internal connectivity ($f_{\text{int}} = 3.0$), synaptic short-term plasticity configured as in Figure 6.16.

parameters Table 6.3 summarizes the set of PCSIM simulation parameters that has been found to result in network dynamics which exhibit a well performing self-stabilization (Bill, 2008).

Description	Parameter	Unit	Value	σ_D/μ_D	s_D/μ_D
Network Architecture					
Number of exc neurons	N_e		144		
Number of inh neurons	N_i		48		
Conn prob from exc to exc neurons	p_{ee}		0.12		
Conn prob from exc to inh neurons	p_{ei}		0.24		
Conn prob from inh to exc neurons	p_{ie}		0.36		
Conn prob from inh to inh neurons	p_{ii}		0.72		
Neurons (Excitatory and Inhibitory)					
Membrane Capacitance	C_m	nF	0.2	0.3	0.5
Reset potential	V_{reset}	mV	-80.0	0.1	0.2
Inhibitory reversal potential	E_i	mV	-80.0	0	0
Leakage reversal potential	E_l	mV	(swept) -62 ... -56		
Firing threshold voltage	V_{thresh}	mV	-55.0	0.01	0.02
Excitatory reversal potential	E_e	mV	0.0	0	0
Leakage resistance	$1/g_l$	MΩ	25.0	1.0	0.8
Refractory period	τ_{ref}	ms	1.0	0	0
Synapses					
Cond amp for exc to exc synapses	g_{ee}	nS	0.35	0.6	0.7
Cond amp for exc to inh synapses	g_{ei}	nS	0.25	0.6	0.7
Cond amp for inh to inh synapses	g_{ii}	nS	1.00	0.6	0.7
Cond amp for inh to exc synapses	g_{ie}	nS	1.50	0.6	0.7
Cond amp for ext to exc synapses	$g_{ext,e}$	nS	(basis value) 6.00	0.6	0.7
Cond amp for ext to inh synapses	$g_{ext,i}$	nS	(basis value) 20.0	0.6	0.7
Cond time constant for all synapses	τ_{syn}	ms	30.0	0.3	0.5
Maximum facilitation factor			1.5	0.5	0.5
Minimum facilitation factor			1.0	0.5	0.5
Facilitation time constant	τ_{fac}	ms	50.0	0.3	0.5
Relative facilitation step per spike			0.27	0.3	0.5
Maximum depression factor			1.0	0.5	0.5
Minimum depression factor			0.7	0.5	0.5
Depression time constant	τ_{dep}	ms	50.0	0.3	0.5
Relative depression step per spike			0.27	0.3	0.5
External Stimulus: Poisson Spike Trains					
Number of exc external spike sources	$N_{ext,e}$		24		
Number of inh external spike sources	$N_{ext,i}$		24		
Conn prob from ext source to neuron	p_{ext}		0.1		
Firing rate per spike train	ν_{ext}	Hz	(uniform) 8 ... 12		
Experiment					
Simulated time per exp run	T_{exp}	ms	5000	(0 ms – 1000 ms not evaluated)	
Number of exp runs per param set	n_{stat}		25		

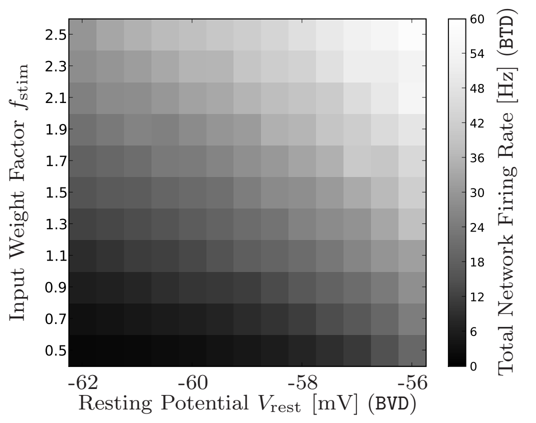
Table 6.3: Full set of PCSIM experiment parameters. All values given in `BVD` and `BTD`.

Simulation Results For the network configurations A, B, C and D, Figure 6.17 shows the resulting total network firing rates ν_{net} as a function of V_{rest} and f_{stim}. In case of weak recurrent synapses (sub-figures **(a)** and **(b)**), the total network firing rate is strongly determined by the two swept parameters across the full range of considered values. For the strong internal feedback (sub-figure **(c)**), the sensitivity is very high, output rates of more than 100 Hz can be observed. For the networks with dynamic synapses (sub-figures **(b)** and **(d)**), a significant firing rate stabilization can only be observed in case of a strong network-internal synaptic feedback (sub-figure **(d)**). For the weak recurrence, the self-tuning effects of the architecture are obviously not strong enough.

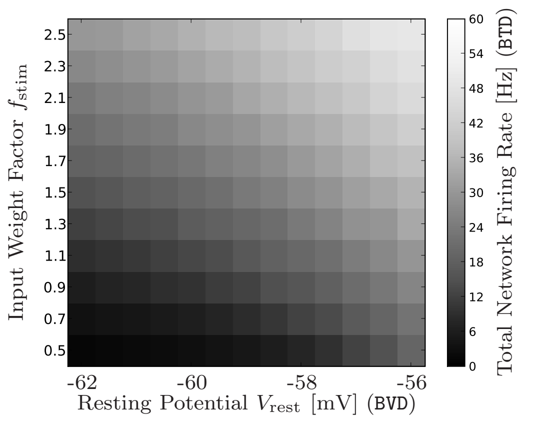
In order to illustrate the functionality of the mutual damping and amplifying, respectively, Figure 6.18 plots the difference between the excitatory and inhibitory population firing rates $\Delta\nu \equiv \nu_{\mathrm{e}} - \nu_{\mathrm{i}}$ for the cases C and D.

The effect is obvious: The firing rate differences $\Delta\nu$ in sub-figure **(a)** (static synapses) are nearly independent of the swept parameters V_{rest} and f_{stim}. But in the case of dynamic synapses (sub-figure **(b)**), the difference between the firing rates of both populations exhibits an obvious dependency on the swept parameters. For a weak external stimulation and a low resting potential (lower left corner), the excitatory population is more active, while for a strong external stimulation and a high resting potential (upper left corner), the inhibitory population dominates the excitatory one. Obviously, the total network firing rate is controlled via this discrepancy in the population firing rates.

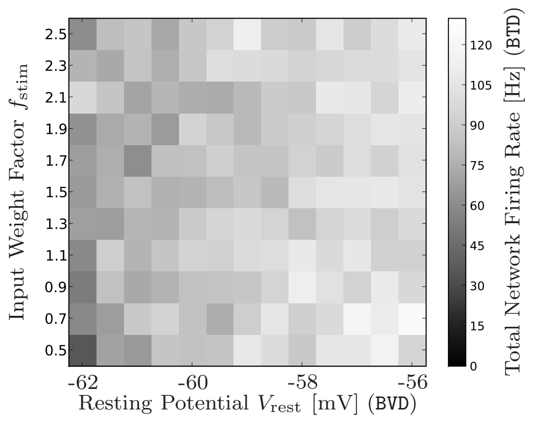
For the presented preparation simulations it can be concluded that, with the chosen parameter values, the described network architecture *does* exhibit self-stabilizing features in terms of its total output firing rate – despite the small number of utilized neurons. Hence, in a next step, the experiment is ported to the `FHW-1` system.



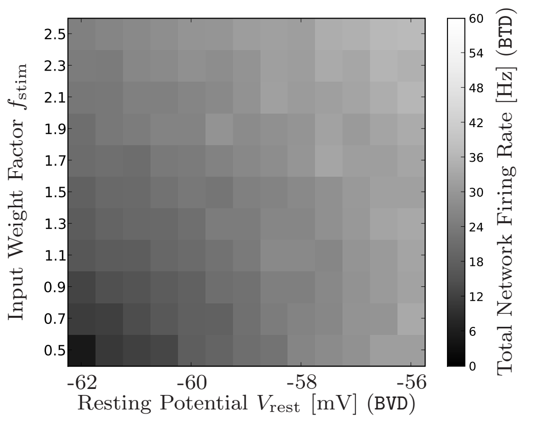
(a) Case A: Internal synapses static and weak.



(b) Case B: Internal synapses dynamic and weak.

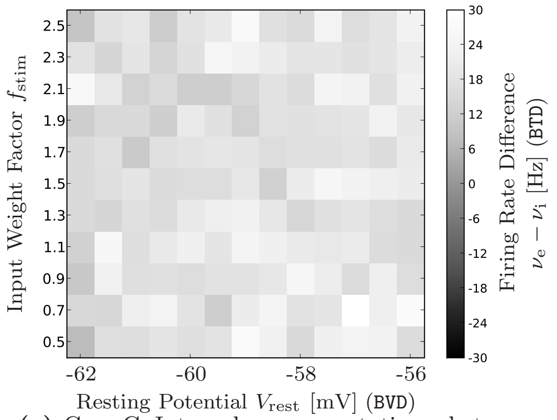


(c) Case C: Internal synapses static and strong.

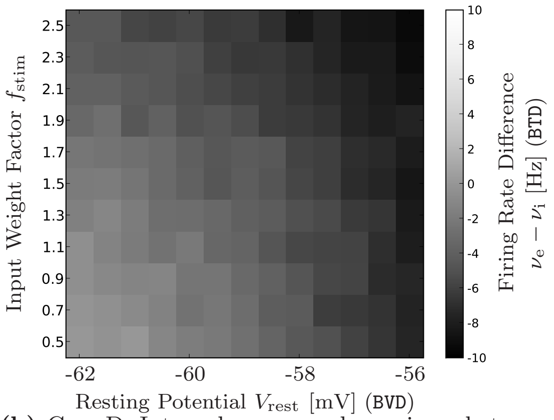


(d) Case D: Internal synapses dynamic and strong.

Figure 6.17: PCSIM Simulation: Total network firing rate as a function of the resting potential V_{rest} of all network neurons and of the stimulation strength factor f_{stim}. The four sub-plots have been acquired with different versions of the network architecture described in the main text: **(a)** Case A, weak recurrent connections, all synapses are static. **(b)** Case B, weak recurrent connections, the internal synapses are depressing and facilitating according to Figure 6.16. **(c)** Case C, strong recurrent connections, all synapses are static. **(d)** Case D, strong recurrent connections, the internal synapses are dynamic according to Figure 6.16. The firing rates in **(a)** and **(b)** are strongly determined by V_{rest} and f_{stim}. The network shown in **(c)** exhibits an even increased sensitivity to the external stimulation (note the different color code). The variant **(d)** exhibits almost stable output firing rates for most of the scanned parameter space. Only for cases of weak stimulation and a low resting potential (lower left corner), the output firing rate is significantly lower.



(a) Case C: Internal synapses static and strong.



(b) Case D: Internal synapses dynamic and strong.

Figure 6.18: PCSIM Simulation: Difference between the firing rates ν_e and ν_i of the excitatory and inhibitory populations in the network as a function of the resting potential V_{rest} of all network neurons and of the stimulation strength factor f_{stim}. The two sub-plots have been acquired with different versions of the network architecture described in the main text: **(a)** Case C, strong recurrent connections, all synapses are static. **(b)** Case D, strong recurrent connections, the internal synapses are dynamic according to Figure 6.16. The firing rate differences in **(a)** are nearly independent of the swept parameters. In the case of dynamic synapses **(b)**, though, the difference between the firing rates of both populations exhibits an obvious dependency on the swept parameters. For weak stimulation and a low resting potential, the excitatory population is more active, while for a strong stimulation and a high resting potential, the inhibitory population dominates. The total network firing rate is controlled via this discrepancy in the population firing rates.

Hardware Realization

Implementing the architecture shown in Figure 6.16 on the hardware system requires a few extra considerations, which are explained in the following.

Handling of Recording Problems Due to the output spike deadlock problem of the `FHW-1.3` system (see Section 4.3.1), only three out of 192 hardware neurons can be arbitrarily recorded. Hence, in order to acquire reliable information about the average firing activity in the two populations of the network, a given setup has to be executed multiple times, while each time different neurons are recorded. The chosen approach is to re-execute the experiment for every individual network configuration $20 \times 5 = 100$ times. Instead of one PCSIM run, i.e. for every randomly determined connection and stimulation pattern, the following is done in hardware: Twenty times, three neurons are chosen randomly and recorded for five repeated executions of the same experiment. This five time repetition is applied in order to minimize the impact of analog noise and various design-related issues (see Section 4.3) onto the recorded firing rates. After every five runs, another random triple of neurons is chosen to be recorded.

Synapse Parameters Due to difficulties in the configuration of the depression and facilitation features of the hardware synapses (see Section 4.3.10), a direct translation between the PCSIM parameters and the hardware characteristics is not possible. Because of the same reason, an analytical conversion between the static and dynamic synapse weights is not possible either. The author of Bill, 2008 used oscilloscope measurements as presented in Figure 6.8 in order to establish appropriate translation factors. Still, the chosen parameter values for the presented hardware experiments result from a merely coarse tuning and do not yet incorporate any optimization effort. The discrete four bit synapse node values determine the weight proportion among the connections: Excitatory to excitatory synapses receive a mean value of $\omega_{\mathrm{ee}} = 7.5$, inhibitory to inhibitory connections are set to $\omega_{\mathrm{ii}} = 5$, and the inter-population connections are set to $\omega_{\mathrm{ei}} = 2.5$ and $\omega_{\mathrm{ie}} = 2.5$. (In Section 3.1.5, the applied technique of mapping continuous weights to the available discrete hardware values is explained.) Furthermore, like in the PCSIM study, all values are chosen from Gaussian distributions around these mean values, with relative standard deviations of $\sigma_\omega = 0.6$ and a relative spread of $s_\omega = 0.7$.

Stimulus Intensity Due to changes in the synapse setup (see above), a re-adjustment of the stimulus is necessary in order to achieve satisfying output firing rates. The number of input channels is increased to 64, which is the maximum number that is possible in case of 192 utilized hardware neurons with full feedback flexibility (compare with Section 6.2.1). 48 of those channels are excitatory, 16 inhibitory. The rate per channel is increased to 12 Hz (`BTD`), which also represents the maximum due to the limited hardware input bandwidth (see Section 6.1.2). Every network neuron (uniformly) samples three to six excitatory and three to six inhibitory spike trains from this pool of externally generated Poisson-type stimuli.

Hardware Results Figure 6.19 shows the total network firing rate acquired on the `FHW-1` system, again as a function of both V_{rest} and f_{stim}. The cases C and D (definition see above) are plotted in sub-figures **(a)** and **(b)**, respectively.

The figure shows that the firing rate self-tuning of the network with the dynamic synapses works well: While in the case of the static synapses (sub-figure **(a)**) the network firing rate is extremely high for nearly all considered parameter combinations, the network with the

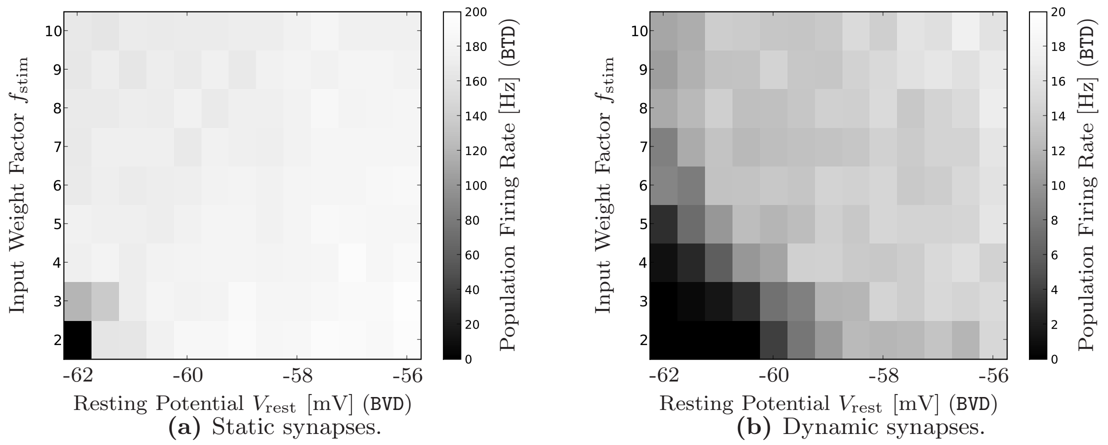


Figure 6.19: Hardware Realization: Total network firing rate as a function of the resting potential V_{rest} of all network neurons and of the stimulation strength factor f_{stim}. The two sub-plots have been acquired with different versions of the network architecture described in the main text: **(a)** Case C, strong recurrent connections, all synapses are static. **(b)** Case D, strong recurrent connections, the internal synapses are dynamic according to Figure 6.16. The network shown in **(a)** exhibits a strong sensitivity to the external stimulation (note the different color code). For nearly all considered parameter sets, the firing activity saturates at an upper limit of $\nu_{\text{max}} \approx 200\,\text{Hz}$ (BTD). The variant **(b)** exhibits a almost stable output firing rates of approximately 15 Hz (BTD) for most of the scanned parameter space. Only for cases of weak stimulation and a low resting potential (lower left corner), the output firing rate is significantly lower.

dynamic synapses (sub-figure **(b)**) exhibits a stable rate of $\nu_{\text{net}} \approx 15\,\text{Hz}$ (BTD) for most of the covered parameter space.

In Figure 6.20, the same firing rates like in Figure 6.19 are plotted, although this time not as a function of V_{rest} and f_{stim}, but rather versus the firing rate standard deviation over the 20 runs that are acquired per parameter set. This is done for both the static and the dynamic synapse setups. The plot illustrates that for the dynamic architecture (grey triangles) not only the mean firing rate over multiple repetitions is remarkably stable, but that such networks also exhibit a high output rate robustness against varying random connections and stimuli based on the same statistics. The absolute variations from run to run are much larger in the case of the static architecture.

Conclusions Drawn From This Section

It can be concluded that the short-term plasticity features implemented in the FHW-1 system provide a possibility to stabilize network dynamics against variations in both the applied stimulation intensity and the responsiveness of the utilized neurons. This can be used to minimize the distorting effects of unwanted hardware-specific variations e.g. in the firing sensitivity of the neuron circuits (see Section 5.2.5). The realization of the described self-balancing paradigm on the hardware system revealed a variety of technical difficulties and problems (see e.g. Section 4.3.10), the most of which still have to be solved. How much information processing properties are left in such a modulated network remains an open question. This has to be investigated in further studies.

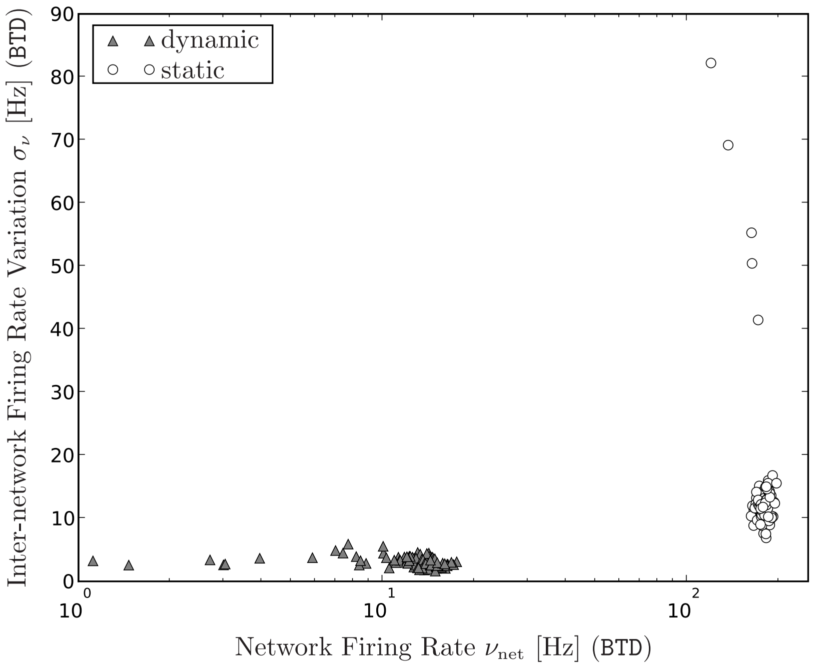


Figure 6.20: For every tested parameter set, the mean value ν_{net} (horizontal) and the standard deviation σ_ν (vertical) over the firing rates acquired in 20 repetitions of the corresponding experiment – with new random connections and stimuli each – are plotted as one data point. The white circles represent the runs with static synapses (compare with Figure 6.19 **a**), the grey triangles represent the experiments performed with dynamic synapses (compare with Figure 6.19 **b**).

Conclusion and Outlook

This thesis aims at the establishment of a novel modeling tool within neuroscience, namely a specific kind of neuromorphic hardware system. For this purpose, a comprehensive study was presented that incorporates the development of a novel methodological framework, the creation of software and the design and execution of experiments.

An existing prototype device of the considered neuromorphic system was put into operation. The chip features were extensively tested, and a specification of process-inherent imperfections and prototype- or design-related properties was extracted. These steps were essential preconditions for further developments: A novel methodological framework was presented that incorporates paradigms for a biological interpretation of the hardware output, calibration algorithms and techniques for the handling of device-specific issues. An advanced software interface was developed that implements the suggested methods and provides the usability for non-hardware-experts. Calibration results were presented that prove the functionality of the chosen hardware tuning approaches. The thesis illustrated the necessity of these calibration steps and documented the successful performance of biologically inspired neural network experiments. The concluding remarks given in the following provide a differentiated answer to the central question of this dissertation:

Can neuromorphic hardware devices of the investigated type serve as modeling tools for neuroscience and provide new insights into neural information processing?

Modeling with the `FHW-1` System Before this question is discussed for the wafer-scale device that is currently under development (see Section 2.2), an answer is given with respect to the chip-based prototype `FHW-1` system utilized throughout this thesis (see Section 2.1). For various reasons, no significant neuroscientific insights can be expected from the utilization of this system in its current setup. The high emulation speed of the `FHW-1` device cannot compensate for its small number of neurons and the connectivity restrictions, especially since the currently applied communication network between the neuromorphic chip and the controlling host computer significantly slows down the work flow (see e.g. Section 6.2.1). As a further design-related drawback, the bandwidth for feeding stimulation events into the chip and reading back output spikes imposes relevant limits (see Section 4.3.7). These kinds of constraints are inherent to the prototype nature of the used device and will be overcome by the wafer-scale `FHW-2` system.

A different type of issues are implementation-specific imperfections, which can be solved by minor revisions of the `FHW-1` design. Caused by such flaws, the configurable values of certain prototype parameters have been shown to be insufficient or unreliable, see e.g. Section 4.3.2, Section 4.3.4 and Section 4.3.10.

The experiments presented in Section 6.2 proved that even with the utilized prototype hardware system cortically inspired network architectures and dynamics can be generated. Despite the small number of applied neurons and a set of technical obstacles imposed by the early development status of the device (e.g. the spike recording problems explained in Section 4.3.1 or the unreliable excitatory reversal potential described in Section 4.3.4), a remarkable qualitative and partly even quantitative correspondence between the hardware emulations and the output of reference software simulations could be achieved. Generally, there is no sense in aiming at a *perfect* quantitative matching between a software simulator and a neuromorphic hardware system. Since analog or mixed-signal devices will never generate a totally reproducible output, they are not suitable for tasks that require perfectly precise parameter values and ideal constituents. But since biological neural systems are subject to similar variations, models of cortical architectures that rely on an ideal substrate are implausible.

Modeling with the `FHW-2` System Considering the promising results and the knowledge acquired with the prototype `FHW-1` chip, the fact that most of the presented methods and higher-level software modules are directly transferable to the wafer-scale `FHW-2` system suggest that this currently developed device will actually provide a tool with which large-scale cortical models can successfully be implemented. Among a variety of possible configurations, architectures will be realizable that – on a single wafer device – resemble approximately 1 mm^3 of mammalian cortical tissue: It will be possible to set up networks in the order of 10000 neurons, each receiving 1000 to 5000 synaptic inputs from other neurons within the network. The dynamics of such networks will be emulated with a speedup factor of approximately 10^4, i.e. one second of chip time will generate the output corresponding to more than two hours of biological time.

This acceleration factor, combined with the programmability of the synaptic connections, the configurability of the implemented long-term and short-term synaptic plasticity mechanisms and the spike-frequency adaptation feature, can be exploited in a variety of possible experimental scenarios (see Section 3.1.1). The exploration of dynamics of a macroscopic fraction of the cortex will be possible in terms of large parameter space scannings, in long-term learning or self-organization studies or within an interactive, intuition-guided scheme. All kind of systematic investigations will be feasible to an extent and with a large sample statistics that is unapproachable by software simulations.

This does not imply that such neuromorphic hardware systems can *replace* software simulations. The physical nature and the slow production cycles of the devices impose a very limited flexibility in terms of the implemented constituents. To quickly add a new feature to the neuron or synapse models within a network and to test the impact of this change onto the emerging network dynamics is a freedom that is reserved to software simulations or to programmable devices like FPGAs[2]. Instead, neuromorphic devices will provide a *complementary* research tool for the systematic or interactive parameter exploration of large architectures that consist of established neuron and synapse models.

Requirements and Challenges A further question stated in the introduction of this thesis relates to the *requirements for the acceptance of such neuromorphic tools in the established modeling communities*. The experience acquired with the work on the `FHW-1` system raised the awareness of some important conditions. It is unlikely that a neuromorphic system will be adopted by modelers if its operation requires an intensive study of hardware-specific details or interfacing techniques. The presented integration of the hardware interface into the PyNN framework addresses this aspect in an elegant way (see Section 3.1.3 and Section 3.2).

[2]Field Programmable Gate Array

Another important requirement for the acceptance of such a system is a certain standard of configuration reliability. Unstable reversal potentials as described in Section 4.3.2 and 4.3.4 impede the transfer of existing experiments to the hardware substrate and have no obvious biological correspondence. More generally speaking, the modeling community must be provided with a *calibrated* hardware system that has *well specified* remaining inhomogeneities, but that is free of practically unpredictable mechanisms that significantly affect the model dynamics.

The initially stated question on *the technical challenges and obstacles on the way towards neuroscientific modeling with neuromorphic hardware* also refers to this essential requirement of a calibratable hardware device. With chip technologies as utilized for the `FHW-1` and the `FHW-2` system, it is impossible to produce a neuromorphic device of comparable complexity that can be reasonably operated in an uncalibrated state. Therefore, a set of appropriate tuning parameters and monitoring possibilities has to be provided by every hardware design that allows to counterbalance the unavoidable functional distortions caused by transistor-level substrate fluctuations. In Section 5.2, the calibration of the `FHW-1` chip was demonstrated, and e.g. the highly laborious determination of PSP integrals for the purpose of synapse driver tuning illustrates the obstacles that can arise from insufficient monitoring opportunities. In order to minimize the additional chip area and communication infrastructure consumed by such tuning opportunities, they have to be economically deployed. The following principles are considered essential in this context:

- The relative proportions of the chip-intrinsic time constants are more important than their absolute values, since the applied speedup factor is a free parameter. The available ranges of time constants of membranes, synaptic conductance courses, short-term and long-term synaptic plasticity need to be well balanced. Therefore, if not configurable individually, at least each of those groups should be adjustable in a wide range.
- At least two arbitrarily adjustable groups of synaptic time constants are highly desirable in order to implement different excitatory and inhibitory efficacy periods.
- All modules within a device that provide input for *multiple* other modules, as e.g. the synapse drivers in the `FHW-1` system, need to be individually configurable, because otherwise in case of a malfunction all dependent modules will be affected as well.
- Since an excessive firing behavior of individual neurons can distort the dynamics of a whole recurrent network, every neuron should have a free tuning parameter for its responsiveness, e.g. the firing threshold.

A further opportunity is not essential but probably very useful: For a very small subset of neuronal and synaptic circuits, all dynamically evolving variables and static parameter values could be made accessible. Such testing units could be used to acquire a deeper understanding of possibly unexpected chip-specific dynamics, and a variety of scenarios can be tested with full transparency.

Not only the remaining inhomogeneities of a calibrated device need to be specified. The information about all distortions introduced by the mapping of a given model to the hardware substrate have to be transparently accessible to the user. This refers to parameter value changes necessary due to e.g. low hardware value resolutions as well as to changes in the connectivity structure due to limited routing possibilities (see Section 3.1.6).

Since the speed of the neuromorphic system is essential to its attractiveness for modelers, the mentioned mapping process and the transfer of configuration data to the system need to be as fast as possible. A high-performance link between the controlling host PC and the system is inevitable, and all involved software layers must be optimized with respect to a rapid transmission of configuration, stimulation and output data (see the wafer-scale support hardware described in Section 2.2 and the software framework outlined in Section 3.2.2).

All the requirements mentioned up to this point are about to be met with the currently developed `FHW-2` system and the corresponding software framework. With the novel operation paradigms presented in this dissertation and building upon the described case studies, it can be concluded that supposably soon the communities of neuromorphic engineering and computational neuroscience can start to mutually benefit from each other.

Perspectives

The following paragraphs give an impression of the ongoing and planned work with the `FHW-1` system and outlines a selection of prospects concerning the deployment of the wafer-scale `FHW-2` system.

Open Issues for the `FHW-1` System

As indicated in this thesis, a set of open issues needs to be solved for the `FHW-1` system, the most important of which will be recapitulated here. Since substantial parts of the `FHW-1` circuitry will be incorporated in the `FHW-2` system, too, solving all known problems of the prototype chip is an essential precondition for the successful operation of the wafer-scale device.

The `FHW-1` design needs a revision in order to solve some critical malfunctions described in Section 4.3. The most important issues are the spike recording problems (Section 4.3.1), the instability of the excitatory reversal potential (Section 4.3.4) and the interdependency between the firing threshold precision and the effective reset potential (Section 4.3.3). The disproportionality of chip-intrinsic time constants (Section 4.3.5) should also be addressed in such a revision. The precise reason for the parasitic resting potential offsets (Section 4.3.3) needs to be clarified and resolved. The limited range of programmable voltages (see Section 4.3.10) has to be improved. Furthermore, the problems with controlling the STDP functionality have to be further investigated, which possibly requires additional design modifications.

Once the reliability of these hardware parameters is improved, systematic measurements have to be performed in order to determine the relation between applied hardware parameter values and the resulting biological variables. For example, the range of synaptic time constants that can be adjusted via the synapse driver decay current $I^{\mathrm{ctrl}}_{\tau_{\mathrm{fall}}}$, which currently has to be kept at a fixed value in order to minimize the resting potential distortions, has to be investigated. The extracted relation between applied hardware values and observed time constants has to be incorporated into the software flow.

Due to a lack of experience with the underlying mechanisms, the software framework presented in Section 3.2.2 provides only insufficient control mechanisms for the `FHW-1` long- and short-term synaptic plasticity features. In order to improve this, systematic measurements and further investigations of these mechanisms are necessary.

Self-Organizing Winner-Take-All Architectures in Hardware

The goal of the self-organization experiments outlined in the following is to start utilizing the `FHW-1` STDP functionality for tasks that can be varied in their complexity. This will allow for a step-wise plasticity testing and specification procedure. Due to problems with the control of the hardware STDP modules mentioned in Section 4.3.11, the realization of such experiments on the hardware system is still work in progress.

Under the supervision of the author, the implementation of self-organizing winner-take-all classifiers on the `FHW-1` system has been prepared in (Kaplan, 2008). Inspired by the work presented in (Häfliger, 2007), so-called *cross-inhibition* architectures[3] have been set up both with the `FHW-1` chip and in NEST. The quality of the cross-inhibition performance as a function of various parameters has been investigated, and first self-organizing experiments based on synaptic modification via STDP (see Section 2.1.3) have been performed in NEST (see example in Figure 6.21). Four neurons, which strongly inhibit each other, receive the same set of 32 input spike trains via randomly initiated excitatory synapses. In the most simple setup, four possible input classes shall be separated by the network: If a pattern is applied to the input of all four cells, only one neuron is supposed to fire. An input pattern manifests as a period of 1000 ms length during which eight out of the 32 input sources fire with high frequencies while the other input channels remain passive (see sub-figures **a**, **b** and **c**). Sequences of random patterns are applied to the four classifier neurons. During this procedure, the weights of the excitatory input synapses evolve according to an STDP modification rule. After a certain number of cycles (a cycle is a sequence of four patterns), the automatic weight changes lead to an improvement in the *selective* response of the neurons to the applied input patterns.

Comparing STDP Architectures Implemented on Different Neuromorphic Systems

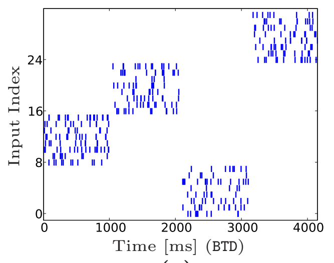
As introduced in Section 1.4, two neuromorphic hardware platforms are under development within the FACETS research project. One type has been described and utilized throughout this thesis, the second type is developed in the Laboratoire IMS at ENSEIRB[4] in Bordeaux (Tomas et al., 2006). The Bordeaux system implements six neurons per chip with a high level of detail in the implemented Hodgkin-Huxley model (Hodgkin and Huxley, 1952). The system is operated in real-time, and an off-chip digital logic provides STDP dynamics for the synapses. As a PyNN interface for that hardware system is currently under development, comparison experiments according to Daouzli et al., 2008 are planned to be performed both on the Bordeaux system and on an `FHW-1` or `FHW-2` device. This would be the first time that a neural network experiment described with PyNN is transfered between two neuromorphic hardware platforms.

The FACETS Demonstrator Project

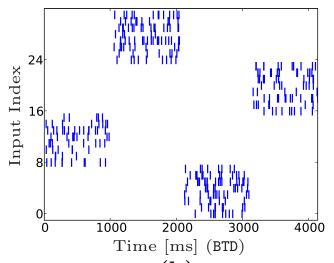
One major goal of the FACETS research project (see Section 1.4) is to coordinate and bundle the efforts of the multi-disciplinary partner groups in order to investigate brain-like computing principles. An experiment that is currently under development and that manifests a collaborative effort is the so-called FACETS *Demonstrator*.

[3]networks in which each neuron inhibits the activity of all other neurons

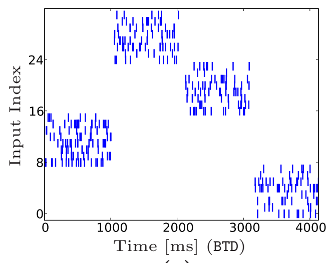
[4]Ecole Nationale Supérieure d'Electronique, Informatique et Radiocommunications de Bordeaux, France



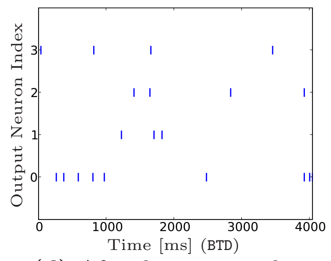
(a)



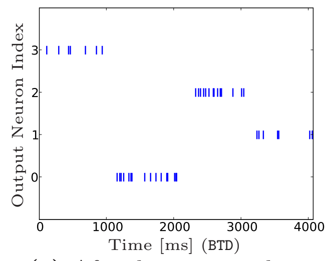
(b)



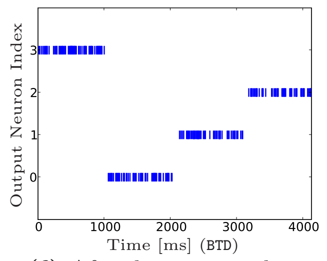
(c)



(d) After learning cycle 0



(e) After learning cycle 13



(f) After learning cycle 99

Figure 6.21: The figures show the self-organized optimization of a winner-take-all classifier implemented in NEST. Input patterns of 32 spike trains each are applied to the four classifier neurons. The sub-figures **a**, **b** and **c** represent the input that is applied during cycles 0, 13 and 99, respectively. (A cycle is a sequence of four different patterns.) During the 1000 ms (BTD) of one pattern, only eight out of the 32 inputs are firing at a time. The classifier neurons are connected to each other with inhibitory synapses. The excitatory input weights evolve according to an STDP modification rule. After multiple cycles of input patterns have been applied, each of the four neurons has learnt to selectively respond to one class of input stimuli. The sub-figures **d**, **e** and **f** show the corresponding classifier output during cycles 0, 13 and 99, respectively. Figure by B. Kaplan.

The Demonstrator is a cortical neural network model emulated with the FHW-2 system. This model will exhibit a specific functionality that can be demonstrated and quantified. In order to provide a transfer of this experiment to established software simulators for verification and performance evaluation, it will be described in the simulator-independent modeling language PyNN (see Section 3.1.3). According to the current status of planning, the network will implement a functional structure of the visual cortex, e.g. a layer II/III attractor memory model as presented in (Johansson et al., 2006).

One important aspect of the Demonstrator approach is the transparent and efficient mapping of a given network model to the limited resources of the hardware substrate (see Section 2.2). The GraphModel mapping tool introduced in Section 3.1.6 is currently being extended with features that allow to automatically extract a PyNN description of the model *after* it has been distorted in order to fit the hardware constraints. Based on this technique, software simulations can analyze both the ideal behavior of a model and the degradation induced by the mapping process. The intention of the Demonstrator participants is to automate this process of mapping evaluation, such that free mapping parameters can be iteratively optimized. Once a mapping procedure has been found that does not significantly degrade the functionality of the architecture, the advantages of the hardware system (see Section 3.1.1) will be exploited for a systematic parameter optimization.

Studying the Possible Transfer of Neural Self-Tuning Features to Electronic Substrates

The following possible application of the FHW-2 system addresses the unreliable production processes of micro- and nano-electronic structures. All current and future information processing devices based on such technologies are affected by this issue, and various strategies to cope with this phenomenon have been developed: On the level of analog circuitry, dedicated design modifications like adaptive amplifier biasing or compensating dummy structures help to reduce the distorting impact of transistor level fluctuations (see e.g. Sansen, 2006). On the functional level, the established digital information processing paradigm introduces a high degree of tolerance against imperfections of electronic variables (Dally and Poulton, 1998). On the device level, designers aim at small die sizes in order to reduce the probability of fatal errors per chip. Every device is individually tested, and for VLSI modules (like processors or memory devices) that are to be embedded in a computer built according to the Turing paradigm, a single production error on the chip usually makes it unusable.

Like human engineered CMOS transistors or nano-devices, the phenotypic constituents of biological nervous systems are subject to variations and imperfections as well. Experimental findings about the dynamical properties of neuron membranes, synaptic connections and neural architectures, some of which have been in focus of this dissertation already, suggest that a variety of mechanisms increase the robustness of the performed computations against substrate variations: Single cell membranes change their integrative properties depending on the intensity of synaptic stimulation (see Destexhe and Pare, 1999 and Section 4.1.2). They can serve as integrators or as coincidence detectors, and the resulting possible non-monotonous input-output relationship can help to establish stable points of network activity (Kumar et al., 2008). Cortico-cortical feedforward-inhibition connections help to control impact periods of thalamo-cortical stimuli (Wielaard et al., 2001). As has been demonstrated in Section 6.2.2, networks of neurons can exploit synaptic depression and facilitation mechanisms to establish a self-balanced network activity that is very robust against parameter and stimulus fluctuations (see Sussillo et al., 2007 and Bill, 2008). Furthermore, the synaptic self-modification based on the temporal correlation between pre- and post-synaptic activity, which is a Hebbian learning principle, has been found to induce adaptive self-optimization of working points in neural architectures (see e.g. Guyonneau et al., 2005). Population codes reduce the distorting impact of fluctuations in single cell properties and thereby increase the reliability of signals that need to be precise, e.g. in the context of motor control (see e.g. Georgopoulos et al., 1986). The list could be continued with brain regions that modulate other regions or even with evolutionary aspects.

The FHW-2 system comprises conductance-based synapse models, a spike-frequency adaptation mechanism, long- and short-term synaptic plasticity features, a large number of neurons, a highly programmable connectivity structure and an intrinsic speed of modeling. Therewith, it provides a platform that allows to systematically investigate these and probably more strategies that are known or assumed to be used by biological neural systems to optimize their working point or to minimize the distorting impact of constituent variations. Since the neuromorphic device is subject to substrate imperfections itself, analyzing the possibilities of transferring such strategies to micro- or nano-scale constituents might provide important methods and insights which help to overcome the increasing problem of unreliable process technologies for the design of future information processing systems.

A Appendix

A.1 Simulation and Emulation Parameters

Unless otherwise expressly mentioned, all hardware experiments and all NEST simulations presented in this thesis have been set up with the parameters listed in the following.

Neuron Parameters

Neuron Parameters			
Description	Parameter	Unit	Value
Membrane capacitance	C_m	nF	0.2
Membrane leakage conductance	g_l	nS	40.0
Reset potential	V_reset	mV	-80.0
Inhibitory reversal potential	E_i	mV	-80.0
Leakage reversal potential	E_l	mV	-75.0
Firing threshold voltage	V_thresh	mV	-55.0
Excitatory reversal potential	E_e	mV	0.0

Table A.1: Neuron default parameters, all given in `BVD` and `BTD`.

Synapse Parameters

Synapse Parameters			
Description	Parameter	Unit	Value
Synaptic CC decay time constant	τ_syn	ms	30.0
Excitatory synaptic CC amplitude	$g_\mathrm{e}^\mathrm{max}$	nS	1.175
Inhibitory synaptic CC amplitude	$g_\mathrm{i}^\mathrm{max}$	nS	7.500

Table A.2: Synapse default parameters, all given in `BVD` and `BTD`.

A.2 Source Code, Documentation and Licences

This section provides information about the location of the source code for all software modules described in this thesis. The access to most of the code repositories listed below is limited to members of the FACETS research project or to members of the Electronic Vision(s) group at the Kirchhoff Institute for Physics in Heidelberg, Germany. Please contact the author (bruederlekip.uni-heidelberg.de) if you are not authorized but require a check-out of the source code.

PyNN

A download link for the PyNN software (not including the hardware specific module) and the complete documentation can be found here:
`http://www.neuralensemble.org/PyNN`

PyNN.hardware.stage1

The most recent version of the hardware specific PyNN module described in Section 3.2.2 is available from:
`https://www.kip.uni-heidelberg.de/repos/FACETSHDDD/software/trunk`

The documentation to the software can be found here:
`https://www.kip.uni-heidelberg.de/repos/FACETSHDDD/software/trunk/doc`
This folder includes the configuration file that is needed by the *doxygen* tool for an automatic extraction of a complete source code documentation. The extraction of both a LaTeX and an HTML documentation is initiated by changing to the local checkout of that folder and calling

```
$ cd doxygen
$ doxygen PyHAL.Doxyfile
```

Analog Unit Tests

All Analog Unit Tests described and listed in Section 3.2.5 can be found here:
`https://www.kip.uni-heidelberg.de/repos/VISION/project/facets/scripts/python/exp/pyNN/test_hardware/AUT`

Calibration Framework

The calibration framework described in Section 5.2 can be found here:
`https://www.kip.uni-heidelberg.de/repos/VISION/project/facets/scripts/python/exp/pyNN/calib_hardware`

Analysis Tools

The NeuroTools software package described in Section 3.2.3 can be found here:
`http://www.neuralensemble.org/NeuroTools`

Software Licenses

- All components of PyNN (see Section 3.1.3) which can be downloaded from the PyNN project homepage (PyNN, 2008) are published under the CeCILL licence (CeCILL 2009).
- The NeuroTools software collection (see Section 3.2.3) is published under the GPL2 licence (GPL 2009).
- Until not officially published, the copyright of the complete hardware-specific software framework presented in Section 3.2 and the calibration framework presented in Section 5.2 is owned by the University of Heidelberg, Germany, and the Technical University of Dresden, Germany.

A.3 Workstation Information

The following hardware systems, support components and calibration data sets have been utilized for the experiments described in this thesis.

Workstation `FHW-1.2-No.5`

System `FHW-1.2-No.5`		
Component	Identification No.	Type
NN Chip	5	Stage 1 version 2
Backplane	n.a.	First generation
Host computer	evolver12	Intel Pentium 4, CPU 2.40GHz, Mem 1555540 kB
Darkwing board	11	
Nathan board	5	
Oscilloscope	facetsscope2	

Calibration Data

Available at https://www.kip.uni-heidelberg.de/repos/FACETSHDDD/software/trunk/src/hardware/stage1/config/calibration, REVISION 6635

Calibration Type	Filename
Output pins	`calibOutputPins_fhws1v2_chip5_setup001.dat`
Parameter `icb`	`calibICB_fhws1v2_chip5_setup001.dat`
Parameter `ileak`	`calibTauMem_fhws1v2_chip5_setup001.dat`
Parameters `drviout` and `drvifall`	`calibSynDrivers_fhws1v2_chip5_setup001.dat`
Synaptic weights	`calibVthresh_fhws1v2_chip5_setup001.dat`

Table A.3: System components and calibration data for the hardware workstation `FHW-1.3-No.5`.

Workstation `FHW-1.3-No.17`

System `FHW-1.3-No.17`		
Component	Identification No.	Type
NN Chip	17	Stage 1 version 3
Backplane	n.a.	Second generation
Host computer	evolver14	Intel Pentium 4, CPU 2.40GHz, Mem 1555540 kB
Darkwing board	16	
Nathan board	2	
Oscilloscope	facetsscope1	

Calibration Data Available at https://www.kip.uni-heidelberg.de/repos/FACETSHDDD/software/trunk/src/hardware/stage1/config/calibration, REVISION 6635	
Calibration Type	Filename
Output pins	`calibOutputPins_fhws1v3_chip17_setup002.dat`
Parameter `icb`	`calibICB_fhws1v3_chip17_setup002.dat`
Parameter `ileak`	`calibTauMem_fhws1v3_chip17_setup002.dat`
Parameters `drviout` and `drvifall`	`calibSynDrivers_fhws1v3_chip17_setup002.dat`
Synaptic weights	`calibVthresh_fhws1v3_chip17_setup002.dat`

Table A.4: System components and calibration data for the hardware workstation `FHW-1.3-No.17`.

Workstation `FHW-1.3-No.18`

System `FHW-1.3-No.18`		
Component	Identification No.	Type
NN Chip	18	Stage 1 version 3
Backplane	n.a.	First generation
Host computer	evolver12	Intel Pentium 4, CPU 2.40GHz, Mem 1555540 kB
Darkwing board	11	
Nathan board	0	
Oscilloscope	facetsscope2	

Calibration Data Available at https://www.kip.uni-heidelberg.de/repos/FACETSHDDD/software/trunk/src/hardware/stage1/config/calibration, REVISION 6635	
Calibration Type	Filename
Output pins	`calibOutputPins_fhws1v3_chip18_setup002.dat`
Parameter `icb`	`calibICB_fhws1v3_chip18_setup002.dat`
Parameter `ileak`	`calibTauMem_fhws1v3_chip18_setup002.dat`
Parameters `drviout` and `drvifall`	`calibSynDrivers_fhws1v3_chip18_setup002.dat`
Synaptic weights	`calibVthresh_fhws1v3_chip18_setup002.dat`

Table A.5: System components and calibration data for the hardware workstation `FHW-1.3-No.18`.

A.4 Practical Recommendations for the `FHW-1` Operation

The following technical recommendations are provided for the operation of all `FHW-1.3` devices:

- Never record from neurons that are not listed as *recordable* (see Section 4.3.1) in the corresponding workstation information file (see Section 3.2.4). Only in the case that not more than one cell per neuron block (64 adjacent circuits, see Section 2.1) is to be recorded, the index is freely selectable.
- It is highly recommended to apply the calibration data that is available for a specific chip.
- The value of the parameter $I_{\tau_{\mathrm{fall}}}^{\mathrm{ctrl}}$ should not exceed $0.15\,\mu\mathrm{A}$ (`HVD`, see Section 4.3.3).
- The bias values for the programmable voltage generators should not exceed $0.02\,\mu\mathrm{A}$ (`HVD`, see Section 5.1.5).
- For a reasonable configuration of the short-term synaptic plasticity mechanism, all values of V_{fac} should be shortened to ground using the $I_{\mathrm{b}}^{\mathrm{test}}$ pin (see Section 5.1.4). V_{max} should be set to the lowest possible value. $V_{\mathrm{synstart}}^{\mathrm{ctrl}}$ is recommended to be set to $0.25\,\mathrm{V}$, V_{dtc} should be $0.7\,\mu\mathrm{A}$ (`HVD`).
- The bias current $I_{\mathrm{syn}}^{\mathrm{bias}}$ for the synapse driver amplitude comparator has been found to be sufficiently large with $1.25\,\mu\mathrm{A}$. The amplitude of the synaptic conductance courses can be sufficiently controlled with a value of $I_{\tau_{\mathrm{rise}}}^{\mathrm{ctrl}} = 0.2\,\mu\mathrm{A}$ (see overshoot problem description in Section 2.1.4, "Synaptic Conductance Courses").
- Install the full operation software package on the host PC that is directly connected with the `FHW-1` system. It is possible to run the higher-level software modules (described in Section 3.2.2) on a computer different from that host PC and communicate with the slow control software via sockets. But at the current status of development, this significantly slows down the communication. Therefore, it is strongly recommended to run all control software directly on the `FHW-1` host PC.

List of Abbreviations

ADC	Analog-to-Digital Converter
aEIF	Adaptive Exponential Integrate-and-Fire (Neuron Model)
AER	Address Event Representation
API	Application Programming Interface
AUT	Analog Unit Test
BTD	Biological Time Domain
BVD	Biological Voltage Domain
CC	Conductance Course
CMOS	Complementary Metal Oxide Semiconductor
DAC	Digital-to-Analog Converter
DLL	Delay-Locked Loop
DNL	Differential Nonlinearity
DTC	Digital-to-Time Converter
EPSP	Excitatory Post-Synaptic Potential
FACETS	Fast Analog Computing with Emergent Transient States
FHW	FACETS Hardware
FHW-1	FACETS Stage 1 Hardware (Chip-Based System)
FHW-2	FACETS Stage 2 Hardware (Wafer-Scale System)
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HICANN	High Input Count Analog Neural Network
HTD	Hardware Time Domain
HTML	Hypertext Markup Language
HVD	Hardware Voltage Domain
I&F Neuron	Integrate-and-Fire Neuron
IPSP	Inhibitory Post-Synaptic Potential
LSB	Least Significant Bit
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
NEST	NEural Simulation Technology, a simulation software
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PCSIM	Parallel neural Circuit SIMulator, a simulation software
PSP	Post-Synaptic Potential
PyHAL	Python Hardware Abstraction Layer
RAM	Random Access Memory
SCSI	Small Computer System Interface
SEM	Standard Error of the Mean
STA	Spike-Triggered Averaging
STDP	Spike-Timing Dependent Plasticity

TDC	Time-To-Digital Conversion
VLSI	Very-Large-Scale Integration
XML	eXtensible Markup Language

Bibliography

M. Abeles. Corticonics: Neural circuits of the cerebral cortex. *Cambridge University Press*, 1991.

D. Abrahams and R. Grosse-Kunstleve. Building hybrid systems with Boost.Python, 2003. URL `http://www.boostpro.com/writing/bpl.pdf`.

D. J. Amit and N. Brunel. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cereb Cortex*, 7(3):237–52, Jan 1997.

J. Anderson, I. Lampl, I. Reichova, M. Carandini, and D. Ferster. Stimulus dependence of two-state fluctuations of membrane potential in cat visual cortex. *Nature Neuroscience*, 3: 617–621, 2000.

L. Badel, W. Gerstner, and M. J. Richardson. Dependence of the spike-triggered average voltage on membrane response properties. *Neurocomputing*, 69(10-12):1062–1065, June 2006.

C. Beaulieu and M. Colonnier. The number of neurons in the different laminae of the binocular and monocular regions of area 17 in the cat, canada. *J Comp Neurol*, 217(3):337–44, Jul 1983.

C. Beaulieu and M. Colonnier. A laminar analysis of the number of round-asymmetrical and flat-symmetrical synapses on spines, dendritic trunks, and cell bodies in area 17 of the cat. *J Comp Neurol*, 231(2):180–9, Jan 1985.

H. K. O. Berge and P. Häfliger. High-speed serial AER on FPGA. In *ISCAS*, pages 857–860. IEEE, 2007.

O. T. Berglihn. RNUM website. `http://rnum.rubyforge.org`, 2006.

G. Bi 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.

E. L. Bienenstock, L. N. Cooper, and P. W. Munro. *Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex*. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6.

J. Bill. Self-stabilizing network architectures on a neuromorphic hardware system. Diploma thesis (English), University of Heidelberg, HD-KIP-08-44, 2008.

T. Binzegger, R. J. Douglas, and K. A. C. Martin. A quantitative map of the circuit of cat primary visual cortex. *J Neurosci*, 24(39):8441–53, Sep 2004.

T. Binzegger, R. J. Douglas, and K. A. C. Martin. Stereotypical bouton clustering of individual neurons in cat primary visual cortex. *J Neurosci*, 27(45):12242–54, Nov 2007.

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

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

J. M. Bower and D. Beeman. *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System (Second edition)*. Springer-Verlag, New York, 1998. ISBN 0387949380.

R. T. Braden. RFC 1122: Requirements for Internet hosts — communication layers, Oct. 1989. URL `ftp://ftp.internic.net/rfc/rfc1122.txt`.

V. Braitenberg and A. Schüz. *Anatomy of the Cortex: Statistics and Geometry.* 1991.

M. Brecht and B. Sakmann. Dynamic representation of whisker deflection by synaptic potentials in spiny stellate and pyramidal cells in the barrels and septa of layer 4 rat somatosensory cortex. *J Physiol (Lond)*, 543(Pt 1):49–70, Aug 2002.

R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94:3637 – 3642, 2005.

R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris Jr, M. Zirpe, T. Natschlager, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. E. Boustani, and A. Destexhe. Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 3(23):349–98, December 2006.

K. L. Briggman and W. Denk. Towards neural circuit reconstruction with volume electron microscopy techniques. *Current Opinion in Neurobiology*, 16(5):562–570, October 2006.

D. Brüderle, A. Grübl, K. Meier, E. Muller, 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)*, volume LNCS 4507, pages 479–486. Springer Verlag, 2007.

D. Brüderle, E. Müller, A. Davison, E. Muller, J. Schemmel, and K. Meier. Establishing a novel modeling tool: A python-based interface for a neuromorphic hardware system. *Front. Neuroinform.*, 3(17), 2009.

N. Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience*, 8(3):183–208, 2000.

A. N. Burkitt, H. Meffin, and D. B. Grayden. Study of neuronal gain in a conductance-based leaky integrate-and-fire neuron model with balanced excitatory and inhibitory synaptic input. *Biological Cybernetics*, 89:119–125, 2003.

N. Caporale and Y. Dan. Spike timing-dependent plasticity: A hebbian learning rule. *Annual review of neuroscience*, February 2008. ISSN 0147-006X.

G. Cauwenberghs. Learning on silicon: A survey. In G. Cauwenberghs and M. A. Bayoumi, editors, *Learning on Silicon: Adaptive VLSI Neural Systems*, pages 1–29, Norwell, MA, 1999. Kluwer Academic Publisher.

CeCILL 2009. Website. `http://www.cecill.info/`.

D. Cohen. Magnetoencephalography: Evidence of magnetic fields produced by alpha-rhythm currents. *Science*, 161(3843):784–786, August 1968.

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

J. Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and B. Linares-Barranco. A spatial contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems. *IEEE Transactions on Circuits and Systems*, 54(7):1444–1458, 2007.

W. J. Dally and J. W. Poulton. *Digital systems engineering.* Cambridge University Press, New York, NY, USA, 1998. ISBN 0-521-59292-5.

Y. Dan and M. Poo. Spike timing-dependent plasticity of neural circuits. *Neuron*, 44(1): 23–30, Sept. 2004.

V. Dante, P. Del Giudice, and A. Whatley. Hardware and software for interfacing to address-event based neuromorphic systems. *The Neuromorphic Engineer*, 2(1):5–6, 2005.

A. Daouzli, S. Saighi, L. Buhry, Y. Bornat, and S. Renaud. Weights convergence and spikes correlation in an adaptive neural network implemented on vlsi. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing (BIOSIGNALS)*, pages 286–291, 2008.

A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger. PyNN: a common interface for neuronal network simulators. *Front. Neuroinform.*, 2(11), 2008.

P. Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* The MIT press, Cambride, Massachusetts, 2001. ISBN 0-262-04199-5.

T. Delbrück and S. C. Liu. A silicon early visual system as a model animal. *Vision Res*, 44 (17):2083–2089, 2004.

A. Destexhe. Conductance-based integrate-and-fire models. *Neural Comput.*, 9(3):503–514, 1997. ISSN 0899-7667.

A. Destexhe and D. Pare. Impact of network activity on the integrative properties of neocortical pyramidal neurons in vivo. *J Neurophysiol*, 81(4):1531–1547, 1999.

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

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

R. Douglas, H. Markram, and K. Martin. *The Synaptic Organization in the Brain*, chapter Neocortex, pages 499–558. Oxford University Press, 5 edition, 2004. ISBN 0-19-515955-1.

R. J. Douglas and K. A. C. Martin. Neuronal circuits of the neocortex. *Annu Rev Neurosci*, 27:419–51, Jan 2004.

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

M. Ehrlich, K. Wendt, and R. Schüffny. Parallel mapping algorithms for a novel mapping & configuration software for the facets project. In *CEA'08: Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications*, pages 152–157, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-960-6766-33-6.

EPFL and IBM. Blue brain project, 2008. URL `http://bluebrain.epfl.ch/`.

J. M. Eppler, M. Helias, E. Muller, M. Diesmann, and M.-O. Gewaltig. PyNEST: a convenient interface to the NEST simulator. *Front. Neuroinform.*, 2(12), 2008.

FACETS. Fast Analog Computing with Emergent Transient States – project website. `http://www.facets-project.org`, 2009.

D. Farina, L. Arendt-Nielsen, R. Merletti, and T. Graven-Nielsen. A spike triggered averaging technique for high resolution assessment of single motor unit conduction velocity changes during fatiguing voluntary contractions. *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, 2:1097–1100 vol.2, 2001. ISSN 1094-687X.

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

J. Fieres, 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.

FreeGLUT. The freeGLUT project website. `http://freeglut.sourceforge.net/`.

Z. Fu, E. Culurciello, P. Lichtsteiner, and T. Delbrück. Fall detection using an address-event temporal contrast vision sensor. In *Proceedings of the 2008 IEEE International Symposium on Circuits and Systems (ISCAS 2008)*, pages 424–427. IEEE, 2008.

R. L. Geiger, P. E. Allen, and N. R. Strader. *VLSI Design Techniques for Analog and Digital Circuits*. McGraw-Hill, 1990.

A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, September 1986. ISSN 0036-8075.

G. Gerstein and B. Mandelbrot. Random walk models for the spike activity of a single neuron. *Biophys J*, 4:41–68, Jan 1964.

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

M.-O. Gewaltig and M. Diesmann. NEST (NEural Simulation Tool). *Scholarpedia*, 2(4):1430, 2007.

K. Glazebrook and F. Economou. PDL: The Perl Data Language. Dr. Dobb's Journal, sep 1997. URL `http://www.ddj.com/184410442`.

GLProgramming. OpenGL programming guide – the redbook. `http://www.glprogramming.com/red/`.

GNU. The make tool. `http://www.gnu.org/software/make/`.

N. H. Goddard, M. Hucka, F. Howell, H. Cornelis, K. Shankar, and D. Beeman. Towards NeuroML: model description methods for collaborative modelling in neuroscience. *Philos Trans R Soc Lond B Biol Sci*, 356(1412):1209–28, 2001.

D. Goodman and R. Brette. Brian: a simulator for spiking neural networks in Python. *Front. Neuroinform.*, 2(5), 2008.

GPL 2009. GNU General Public License 2.0. `http://www.gnu.org/licenses/gpl-2.0.html`.

A. Grübl. personal communication, 2008.

A. Grübl. *VLSI Implementation of a Spiking Neural Network.* PhD thesis, Ruprecht-Karls-University, Heidelberg, 2007. Document No. HD-KIP 07-10.

R. Guyonneau, R. VanRullen, and S. J. Thorpe. Neurons tune to the earliest spikes through stdp. *Neural Computation*, 17(4):859–879, April 2005.

P. Häfliger. Adaptive WTA with an analog VLSI neuromorphic learning chip. *IEEE Transactions on Neural Networks*, 18(2):551–72, 2007.

B. Haider, A. Duque, A. R. Hasenstaub, and D. A. McCormick. Neocortical network activity in vivo is generated through a dynamic balance of excitation and inhibition. *J Neurosci*, 26(17):4535–45, Apr 2006.

A. Hastings. *The Art of Analog Layout.* Prentice-Hall Inc., Upper Saddle River, New Jersey, USA, 2001. ISBN 0-13-087061-1.

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

M. Helias, S. Rotter, M.-O. Gewaltig, and M. Diesmann. Structural plasticity controlled by calcium based correlation detection. *Front. Neuroinform.*, 2(7), 2008.

M. Hines, T. Morse, M. Migliore, N. Carnevale, and G. Shepherd. ModelDB: A database to support computational neuroscience. *Journal of Computational Neuroscience*, 17(1):7–11, 2004.

M. L. Hines and N. T. Carnevale. *The NEURON Book*. Cambridge University Press, Cambridge, UK, 2006. ISBN 978-0521843218.

M. L. Hines, A. P. Davison, and E. Muller. NEURON and Python. *Front. Neuroinform.*, 2009.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117(4):500–544, August 1952. ISSN 0022-3751.

J. D. Hunter. Matplotlib: A 2D graphics environment. *IEEE Computing in Science and Engineering*, 9(3):90–95, 2007.

IEEE. IEEE Token Ring standards. `http://www.ieee802.org/5/`.

INCF Software Database. Website, 2008. URL `http://software.incf.net`.

IST. Information Society Technologies – website. `http://cordis.europa.eu/ist`, 2009.

H. Jaeger, W. Maass, and J. Principe. Special issue on echo state networks and liquid state machines. *Neural Networks*, 20(3):287–289, Apr. 2007.

W. Jin, R. J. Zhang, and J. Y. Wu. Voltage-sensitive dye imaging of population neuronal activity in cortical tissue. *Journal of neuroscience methods*, 115(1):13–27, March 2002.

C. Johansson and A. Lansner. Towards cortex sized artificial neural systems. *Neural Networks*, 20(1):48–61, 2007.

C. Johansson, M. Rehn, and A. Lansner. Attractor neural networks with patchy connectivity. *Neurocomputing*, 69(7-9):627–633, Jan 2006.

R. Jolivet, R. Kobayashi, A. Rauch, R. Naud, S. Shinomoto, and W. Gerstner. A benchmark test for a quantitative assessment of simple neuron models. *Journal of Neuroscience Methods*, 169(2):417 – 424, 2008. ISSN 0165-0270. Methods for Computational Neuroscience.

E. Jones, T. Oliphant, and P. Peterson. SciPy: Open source scientific tools for Python, 2001. URL `http://www.scipy.org/`.

R. Jung, W. Berger, and H. Berger. Fiftieth anniversary of Hans Berger's publication of the electroencephalogram. His first records in 1924–1931 (author's transl). *Arch Psychiatr Nervenkr*, 227:279–300, Dec 1979.

Jungo Ltd. *WinDriver*. 1 Hamachshev Street, P.O.Box 8493, Netanya 42504, Israel, 2007.

B. Kaplan. Self-organization experiments for a neuromorphic hardware device. Diploma thesis (English), University of Heidelberg, HD-KIP-08-42, 2008.

B. Kaplan, D. Brüderle, J. Schemmel, and K. Meier. High-conductance states on a neuromorphic hardware system. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN)*, 2009.

B. W. Kernighan and D. M. Ritchie. *The C Programming Language.* Prentice-Hall International Inc., London, 1978. ISBN 0-13-110163-3.

J. N. Kerr, D. Greenberg, and F. Helmchen. Imaging input and output of neocortical networks in vivo. *Proc Natl Acad Sci U S A*, 102(39):14063–14068, September 2005. ISSN 0027-8424.

D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*, section 6.2.1, pages 409–426. Addison-Wesley, Reading, Massachusetts, third edition, 1997.

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

I. Lampl, I. Reichova, and D. Ferster. Synchronous membrane potential fluctuations in neurons of the cat visual cortex. *Neuron*, 22(2):361–74, Feb 1999.

L. Lamport. *LaTeX – A Document Preparation System.* Addison-Wesley, second edition, 1994. Updated for LaTeXe.

J. Langner. Development of a Parallel Computing Optimized Head Movement Correction Method in Positron Emission Tomography. Master of computer science thesis, University of Applied Sciences Dresden and Research Center Dresden-Rossendorf, 2003. URL `http://www.jens-langner.de/ftp/MScThesis.pdf`.

H. P. Langtangen. *Python Scripting for Computational Science.* Springer, 3rd edition, February 2008. ISBN 3540739157.

LeCroy. X-stream oscilloscopes - remote control manual. Technical Report Revision D, LeCroy Corporation, 700 Chestnut Ridge Road, Chestnut Ridge, NY 10977-6499, 2005. URL `http://lecroygmbh.com`.

R. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, 2007.

R. Legenstein, C. Naeger, and W. Maass. What can a neuron learn with spike-timing-dependent plasticity? *Neural Computation*, 17(11):2337–2382, November 2005.

J.-F. Léger, E. A. Stern, A. Aertsen, and D. Heck. Synaptic integration in rat frontal cortex shaped by network activity. *J Neurophysiol*, 93(1):281–93, Jan 2005.

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

M. A. Lewis, R. Etienne-Cummings, A. H. Cohen, and M. Hartmann. Toward biomorphic control using custom aVLSI chips. In *Proceedings of the International conference on robotics and automation.* IEEE Press, 2000.

N. K. Logothetis, J. Pauls, M. Augath, T. Trinath, and A. Oeltermann. Neurophysiological investigation of the basis of the fmri signal. *Nature*, 412(6843):150–157, July 2001.

M. Lutz. *Programming Python: Object-Oriented Scripting.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001. ISBN 0596000855. Foreword By-Guido Van Rossum.

W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14 (11):2531–2560, 2002.

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

W. Maass, T. Natschläger, and H. Markram. *Computational models for generic cortical microcircuits*, chapter 18, pages 575–605. 2004b.

H. Markram, J. Lübke, and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps. *Science*, 275:213–215, 1997.

H. Markram, 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(9):5323–5328, Apr. 1998. ISSN 0027-8424.

M. Matsumura, 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(23):7757–7767, 1996.

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

C. A. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78:1629–1636, 1990.

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

C. Mehring, J. Rickert, E. Vaadia, S. C. de Oliveira, A. Aertsen, and S. Rotter. Inference of hand movements from local field potentials in monkey motor cortex. *Nat. Neurosci.*, 6(12): 1253–1254, 2003.

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

S. Mitra, S. Fusi, and G. Indiveri. Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Transactions on Biomedical Circuits and Systems*, 3:(1):32–42, 2009.

Morrison, Abigail, Diesmann, Markus, Gerstner, and Wulfram. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, 98(6):459–478, June 2008. ISSN 0340-1200.

A. Morrison, C. Mehring, T. Geisel, A. Aertsen, and M. Diesmann. Advancing the boundaries of high connectivity network simulation with distributed computing. *Neural Comput.*, 17 (8):1776–1801, 2005.

A. Morrison, A. Aertsen, and M. Diesmann. Spike-Timing-Dependent Plasticity in Balanced Random Networks. *Neural Comp.*, 19(6):1437–1467, 2007.

E. Müller. Operation of an imperfect neuromorphic hardware device. Diploma thesis (English), University of Heidelberg, HD-KIP-08-43, 2008.

E. B. Muller. *Markov Process Models for Neural Ensembles with Spike-Frequency Adaptation.* PhD thesis, Ruprecht-Karls University Heidelberg, 2006.

T. Netter and N. Franceschini. A robotic aircraft that follows terrain using a neuromorphic eye. In *Conf. Intelligent Robots and System*, pages 129–134, 2002.

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

NeuroTools. Website. `http://neuralensemble.org/trac/NeuroTools`, 2008.

Nokia. Qt cross-platform application framework. `http://www.qtsoftware.com/`, 2009.

Numpy. Website. `http://numpy.scipy.org`, 2008.

K. Ohki and R. C. Reid. Specificity and randomness in the visual cortex. *Curr Opin Neurobiol*, 17(4):401–7, Aug 2007.

M. Okun and I. Lampl. Instantaneous correlation of excitation and inhibition during ongoing and sensory-evoked activities. *Nat Neurosci*, 11(5):535–7, May 2008.

T. E. Oliphant. Python for scientific computing. *IEEE Computing in Science and Engineering*, 9(3):10–20, 2007.

OpenGL. Website. `http://www.opengl.org`.

B. Ostendorf. Charakterisierung eines Neuronalen Netzwerk-Chips. Diploma thesis (German), University of Heidelberg, HD-KIP 07-12, 2007.

M. Oster, A. M. Whatley, S.-C. Liu, and R. J. Douglas. A hardware/software framework for real-time spiking systems. In *Proceedings of the 2005 International Conference on Artificial Neural Networks (ICANN2005)*, 2005.

L. Paninski. The spike-triggered average of the integrate-and-fire cell driven by gaussian white noise. *Neural Comput.*, 18(11):2592–2616, 2006. ISSN 0899-7667.

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

D. A. Pecevski, T. Natschläger, and K. N. Schuch. PCSIM: A parallel simulation environment for neural circuits fully integrated with Python. *Front. Neuroinform.*, pending publication.

J. P. Pfister and W. Gerstner. Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity. *J. Neuroscience*, 26:9673–9682, 2006.

S. Philipp. *Design and Implementation of a Multi-Class Network Architecture for Hardware Neural Networks.* PhD thesis, Ruprecht-Karls Universität Heidelberg, 2008.

S. Philipp, A. Grübl, K. Meier, and J. Schemmel. Interconnecting VLSI spiking neural networks using isochronous connections. In *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN'2007)*, volume LNCS 4507, pages 471–478. Springer Verlag, Sept. 2007.

D. Plenz and A. Aertsen. Neural dynamics in cortex-striatum co-cultures–ii. spatiotemporal characteristics of neuronal activity. *Neuroscience*, 70(4):893–924, Feb 1996.

M. Purschke, A. Kandasamy, A. Kriplani, R. Lecomte, P. O'Connor, J.-F. Pratte, V. Radeka, D. Schlyer, S. Southekal, S. Stoll, P. Vaska, A. Villanueva, C. Woody, S. Junnakar, S. Krishnamoorthy, S. Shokouhi, R. Fontaine, and V. Dzhordzhadze. The ratcap conscious small animal pet tomography. *IEEE-NPSS Real Time Conference*, 2005.

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

Python. The Python Programming Language – website. `http://www.python.org`, 2009.

S. Ramon y Cajal. *Histologie du Systeme Nerveux de l'homme et des Vertebres*, volume 2. 1911.

S. Ray and U. S. Bhalla. PyMOOSE: interoperable scripting in Python for MOOSE. *Front. Neuroinform.*, 2(6), 2008.

S. Renaud, 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.

G. V. Rossum. *Python Reference Manual: February 19, 1999, Release 1.5.2.* iUniverse, Incorporated, 2000. ISBN 1583483748.

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

M. Rudolph and A. Destexhe. Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies. *Neural Comput.*, 18(9):2146–2210, 2006. ISSN 0899-7667.

B. Sakmann and E. Neher, editors. *Single-channel recording.* Plenum press, 1995.

W. M. C. Sansen. *Analog Design Essentials (The International Series in Engineering and Computer Science).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387257462.

J. Schemmel. personal communication, 2008.

J. Schemmel, K. Meier, and E. Muller. A new VLSI model of neural microcircuits including spike time dependent plasticity. In *Proceedings of the 2004 International Joint Conference on Neural Networks (IJCNN'04)*, pages 1711–1716. IEEE Press, 2004.

J. Schemmel, A. Grübl, K. Meier, and E. Muller. 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.

J. Schemmel, 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.

J. Schemmel, 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.

F. Schürmann, S. Hohmann, J. Schemmel, and K. Meier. Towards an Artificial Neural Network Framework. In A. Stoica, J. Lohn, R. Katz, D. Keymeulen, and R. Zebulum, editors, *Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware*, pages 266–273. IEEE Computer Society, 2002.

SciPy. Website. `http://www.scipy.org/`.

R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, H. K. Riis, T. Delbrück, S.-C. Liu, S. Zahnd, A. M. Whatley, R. J. Douglas, P. Häfliger, G. Jimenez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco. AER building blocks for multi-layer multi-chip neuromorphic vision systems. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1217–1224. MIT Press, Cambridge, MA, 2006.

M. N. Shadlen and W. T. Newsome. The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *J Neurosci*, 18(10):3870–96, May 1998.

M. Shelley, 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.

G. M. Shepherd, editor. *The Synaptic Organization of the Brain*. Oxford University Press, 198 Madison Avenue, New York, New York, 5 edition, 2004. ISBN 0-19-515955-1.

Y. Shu, 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 (32):10388–10401, November 2003.

M. Smith and A. Kohn. Spatial and temporal scales of neuronal correlation in primary visual cortex. *J Neurosci*, 28(48):12591–12603, Nov 2008.

W. R. Softky and C. Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs. *J Neurosci*, 13(1):334–50, Jan 1993.

S. Song and L. F. Abbott. Cortical development and remapping through spike timing-dependent plasticity. *Neuron*, 32(2):339–350, Oct. 2001.

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

A. Stepanyants, J. A. Hirsch, L. M. Martinez, Z. F. Kisvárday, A. S. Ferecskó, and D. B. Chklovskii. Local potential connectivity in cat primary visual cortex. *Cereb Cortex*, 18(1): 13–28, Jan 2008.

A. Stepanyants, L. Martinez, A. S. Ferecskó, and Z. F. Kisvárday. The fractions of short- and long-range connections in the visual cortex. *Proc Natl Acad Sci USA*, Feb 2009.

B. Stroustrup. *The C++ Programming Language*. Addison Wesley Longman, Amsterdam, February 2000. ISBN 0201700735.

M. Summerfield. *Rapid GUI Programming with Python and Qt.* Prentice Hall, 2008. ISBN 0132354187.

D. Sussillo, T. Toyoizumi, and W. Maass. Self-tuning of neural circuits through short-term synaptic plasticity. *J Neurophysiol*, 97(6):4079–4095, 2007.

The NEST Initiative. Website. `http://www.nest-initiative.org`, 2009.

A. M. Thomson and C. Lamy. Functional maps of neocortical local circuitry. *Frontiers in neuroscience*, 1(1):19–42, Nov 2007.

A. M. Thomson and S. Radpour. Excitatory connections between ca1 pyramidal cells revealed by spike triggered averaging in slices of rat hippocampus are partially nmda receptor mediated. *European Journal of Neuroscience*, 3(6):587–601, 1991.

S. Thorpe, A. Delorme, and R. V. Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14:715–725, 2001.

I. Timofeev, F. Grenier, M. Bazhenov, T. J. Sejnowski, and M. Steriade. Origin of slow cortical oscillations in deafferented cortical slabs. *Cereb Cortex*, 10(12):1185–99, Dec 2000.

M. Toledo-Rodriguez, A. Gupta, Y. Wang, C. Z. Wu, and H. Markram. *The handbook of brain theory and neural networks*, chapter Neocortex: Basic neuron types, pages 719–725. The MIT Press,, Cambridge, MA, second edition, 2002.

J. Tomas, Y. Bornat, S. Saighi, T. Levi, and S. Renaud. Design of a modular and mixed neuromimetic ASIC. In *Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems*, pages 946–949, 2006.

O. Torres-Fernández, C. Golgi, and S. Ramón y Cajal. The Golgi silver impregnation method: commemorating the centennial of the Nobel Prize in medicine (1906) shared by Camillo Golgi and Santiago Ramón y Cajal. *Biomedica*, 26:498–508, Dec 2006.

M. Tsodyks 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, Jan. 1997.

A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1937.

D. van Heesch. The doxygen documentation system. `http://www.stack.nl/~dimitri/doxygen`.

M. C. W. van Rossum. A novel spike distance. *Neural Computation*, 13(4):751–763, 2001.

C. van Vreeswijk and H. Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–6, Dec 1996.

J. D. Victor and K. P. Purpura. Nature and precision of temporal coding in visual cortex: a metric-space analysis. *J Neurophysiol*, 76(2):1310–1326, 1996.

T. P. Vogels and L. F. Abbott. Signal propagation and logic gating in networks of integrate-and-fire neurons. *J Neurosci*, 25(46):10786–95, Nov 2005.

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

J. von Neumann. First draft of a report on the EDVAC. Technical report, Moore School of Electrical Engeneering Library, University of Pennsylvania, 1945. Transscript in: M. D. Godfrey: Introduction to "The first draft report on the EDVAC" by John von Neumann. IEEE Annals of the History of Computing 15(4), 27–75 (1993).

S. Waldert, H. Preissl, E. Demandt, C. Braun, N. Birbaumer, A. Aertsen, and C. Mehring. Hand movement direction decoded from MEG and EEG. *J Neurosci*, 28(4):1000–1008, January 2008.

K. Wendt, 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*, pages 189–194, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-960-6766-33-6.

D. J. Wielaard, M. Shelley, D. McLaughlin, and R. Shapley. How simple cells are made in a nonlinear network model of the visual cortex. *J. Neurosci.*, 21(14):5203–5211, 2001.

C. Xu, W. Zipfel, J. B. Shear, R. M. Williams, and W. W. Webb. Multiphoton fluorescence excitation: new spectral windows for biological nonlinear microscopy. *Proc. Natl. Acad. Sci. U.S.A.*, 93:10763–10768, Oct 1996.

T. Yamazaki and S. Tanaka. The cerebellum as a liquid state machine. *Neural Networks*, 20 (3):290–297, Apr. 2007.

R. S. Zucker and W. G. Regehr. Short-term synaptic plasticity. *Annu. Rev. Physiol.*, 64: 355–405, 2002.

Acknowledgment

I want to express my gratitude to all persons who supported this work, especially:

My family.

Karlheinz Meier and Johannes Schemmel for their supportive mentoring and confidence over many years.

Alain Destexhe for assessing this thesis.

All members of the Electronic Vision(s) group for great team work, for many inspiring conversations and for very diligent proof-reading.

The Softies for being my beloved Softies.

The Hardies for not leaving me alone with these crazy Softies.

Johannes Bill, Bernhard Kaplan and Eric Müller for their contributions to this thesis, for their commitment as diploma students and for being great partners in research, cake production, tabletop soccer and boboaaaak!

Mihai Petrovici, Moritz Schilling and Bernhard Vogginger for a lot of fun, for great matches and great pool parties, for piles of LIDL food and for great softieness.

Jens Kremkow for being a very competent, inspiring and humorous collaborator.

Eilif Muller and Andrew Davison for lots of inspiration and for accepting me as a part of their Python movement.

Tobias Harion, Andreas Bauer, Thomas Pfeil and Jens Rasenack for their committed internship work.

Christoph Walz for support, resonance, reliability and media. hia.

Olivier LaSchiazza for teaching me basic wisdoms of life, aii.

Philip Heuser and Dan Husmann for thousands of enjoyed cycling kilometers.

Claudia Wördemann and Christian Koscher for sharing a warm home.