RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Bernhard Vogginger

Testing the Operation Workflow of a Neuromorphic Hardware System with a Functionally Accurate Model

Diplomarbeit

HD-KIP-TODOTODOTODO

KIRCHHOFF-INSTITUT FÜR PHYSIK

Faculty of Physics and Astronomy
University of Heidelberg

Diploma thesis

in Physics

submitted by

Bernhard Karl Vogginger

born in Singen(Hohentwiel), Germany

February 2010

Testing the Operation Workflow of a Neuromorphic Hardware System with a Functionally Accurate Model

This diploma thesis has been carried out by
Bernhard Karl Vogginger
at the

Kirchhoff Institute for Physics

University of Heidelberg

under the supervision of

Prof. Dr. Karlheinz Meier

Testing the Operation Workflow of a Neuromorphic Hardware System with a Functionally Accurate Model

Neuromorphic hardware represents a promising approach for the investigation and reverse-engineering of biological neural networks. This thesis has been carried out as part of an ongoing effort to develop a large-scale highly-accelerated neuromorphic system. By providing a large and diverse configuration space the system offers the possibility of emulating nearly arbitrary neural network architectures. This, however, sets demanding requirements on the operation of the system. In order to test the associated operation workflow and to improve the responsible software layers, a previously developed executable system simulation of the neuromorphic hardware device was significantly enhanced. The focus lied on the development of functionally accurate models of the hardware neurons and synapses for the system simulation, which is intended to reflect the behavior of the physical hardware as precisely as possible. The operational workflow was successfully tested with the system simulation. The enhanced system simulation was a key ingredient in testing and optimizing the operation workflow. Finally, the functionality of the neurmorphic hardware was demonstrated through the successful emulation of three biologically relevant benchmark models.

Test des Arbeitsablaufs der Ansteuerung einer Neuromorphen Hardware mit Hilfe eines Funktionell Korrekten Modells

Neuromorphe Hardware ist ein vielversprechender Ansatz zur Untersuchung und zum Nachbau von biologischen neuronalen Netzen. Diese Arbeit wurde im Rahmen eines Projektes vollzogen, welches die Entwicklung eines großskaligen hochbeschleunigten neuromorphen Systems zum Ziel hat. Zur Emulation neuronaler Netzwerke verschiedenster Art bietet dieses System einen großen Spielraum an möglichen Konfigurationen. Das wiederum stellt hohe Anforderungen an die Ansteuerung des Systems. Um den zugehörigen Arbeitsablauf zu testen und die dafür zuständige Software zu erweitern, wurde eine ausführbare Systemsimulation der neuromorphen Hardware weiterentwickelt. Der Schwerpunkt lag in der Übertragung von Hardware-Neuronen und -Synapsen in funktionelle Modelle für die Systemsimulation, welche das Verhalten der echten Hardware möglichst präzise widerspiegeln soll. Mit Hilfe der Systemsimulation konnte der Arbeitsablauf nicht nur getestet, sondern auch optimiert werden. Durch die erfolgreiche Emulation von drei biologisch relevanten Benchmark-Modellen wurde die Funktionalität der neuromorphen Hardware demonstriert.

Contents

1 Introduction

1.1 Emergence and the Brain

The whole is more than the sum of its parts.

Aristotle is said to have coined this phrase in order to describe an interesting and, at a first glance, somewhat surprising phenomenon that seems prevalent in nature: the emergence of complex systems with novel properties and functionalities from a multitude of simple components.

When the interaction of simple entities creates a new behavior that can not be traced back to the dynamics of a single entity, such a system is called emergent, as new properties arise or emerge from the interplay of its parts. Quite often, such systems are highly complex, such that their behavior can not be easily understood. It took, for example, many centuries for physicists to trace back the empiric, macroscopic laws of thermodynamics to the interaction of the microscopic constituents of matter. On the other hand, some macroscopic phenomena such as high-temperature superconductivity still await their microscopic explanation.

The behavior of an emergent system does not, in general, depend on any single one of its constituent entities, as it is usually robust against the loss of some of its microscopic components. This principle of robustness can be found in various domains at many different scales in nature. The proper functioning of organisms is not impaired by the loss of single cells, nor do phase transitions depend on the exact number of molecules involved.

Ranging from the billion light-year scales of the Cosmic Microwave Background down to the subatomic scales of quark-gluon plasma, science tries to explain the emerging properties of systems at a larger scale by investigating them at a lower scale (e.g. in time or space). This approach lies at the heart of our ongoing effort to understand the world around us. It is not particular to any single branch of science, as it has been successfully applied over a wide range of scientific disciplines, such as genetics (explanation of phenotypes through gene expression), demography (population dynamics arising from interactions among individuals) or physics (large-scale coherent phenomena as a result of quantum interactions), just to name a few.

Biological neural systems occupy a somewhat exceptional position, as they are arguably the most complex systems we have so far encountered. Understanding them is not only a question of academic interest, as it also promises the key to many age-long philosophical questions, such as the problem of selfhood or the mind-body problem. Without the proper scientific approach, such questions are doomed to the field of mere speculation.

It is not purely by chance that the field of neuroscience has become so highly active and productive during the past few decades. With the dawning of the computer era, on the one hand, and with the invention of powerful experimental techniques, on the other, neuroscientists now have the tools to measure and investigate the inner workings of the brain at previously unimagined detail and then try to reproduce their functionality in custom-built artificial neural networks.

Even to the scientifically untrained, the brain is an astonishing device. It allows its possessor to routinely react with amazing speed to highly complex stimuli, for example while driving a car, to store massive amounts of information, most of which can be recalled at will - a lifetime of memories - or to perform highly sophisticated motor tasks, such as juggling or acrobatics. Another one of its key features is its ability to adapt and learn - all the skills described above are the result of some form of training. Ultimately, the human brain produces, at its highest level of emergence, the still mysterious phenomenon we call consciousness, a result of the complex interaction of external stimuli, learned abilities, memories and sophisticated rationality.

Results from neuroscience have a wide range of application beyond appeasing our innate thirst for knowledge. In their ability to solve a large spectrum of complicated problems, humans are far superior to computers or robots, which are usually specialized for a very restricted category of tasks. Also, the brain's ability to learn and adapt still lies far beyond the possibilities of any artificial device. From this point of view, learning about how these paradigms are realized in biological neural networks may prove useful in creating better, more intelligent synthetic agents. On the other hand, a deeper insight into the workings of human brain may also facilitate the treatment of neurodegenerative diseases or mental disorders. Ultimately, neuroscience may provide the knowledge for creating brain-machine interfaces (see e.g. *Mehring et al.* [2003]) or even artificial brains, entailing unforeseeable possibilities for humankind.

1.2 Approaches to Neuroscience

When investigating the functionality of the brain, different approaches can be taken, depending on the level of detail one chooses as a starting point.

In the psychophysics approach, the brain is roughly subdivided into large areas which are responsible for different tasks, e.g. the processing of visual stimuli. These areas can be further subdivided into smaller units, which specialize on various sub-tasks, such as edge detection in the primary visual cortex or color processing in the visual area V4. By studying the effect on a subject's experience or behavior of systematically varying the properties of a stimulus along one or more physical dimensions, psychophysicists try to determine the function and the interconnections of the different *modules* of the brain.

In a more detailed approach one can focus on populations of neurons which lie close together and display similar activity patterns. One would then argue that these populations are assigned to different tasks and would try to explain the various processes in the brain by describing the interactions among such populations. A prominent example of such a population-based approach is the so-called LISSOM model (*Miikkulainen et al.* [2005]), which uses the concept of self-organizing maps to explain the structure of the primary visual system.

A very detailed approach would be to start by modeling single neurons, which are generally agreed upon as being the basic building blocks, or processing units, of the nervous system. This approach can range from specifying the general behavior of neurons and building so-called formal models (see e.g. *Destexhe* [1997]; *Brette and Gerstner* [2005]) to accurately investigating their substructure and molecular mechanisms (see e.g. *Hodgkin and Huxley* [1952]; *Shouval et al.* [2002]; *Kubota and Kitajima* [2007]). In any case, both electrophysiological experiments at such a detail level and reverse-engineering a functional network out of single neurons are difficult tasks, as there are 10^{11} neurons in the human brain, each of them connected to other

neurons through approximately 10^4 synapses (resulting in an average total of 10^{15} synapses in an adult human brain) (*Shepherd* [2004]).

There are two cardinal approaches for tackling this problem. One possibility is to simulate the electrophysiological behavior of single neurons and their pairwise interaction (i.e. synapse dynamics) in a software environment. The computational complexity of this problem scales strongly with the size of the simulated network, the connection density and the electrophysiological detail level, as does the time needed for simulating such a network (*Morrison et al.* [2005]). Apart from optimizing the simulation algorithms, the only solution for simulating large neural networks, possibly even on a brain scale, lies in increasing the available raw computational power. A prominent example for this paradigm is the Blue Brain Project (*EPFL and IBM* [2008]), which at its current state of development uses over 4 million processors to simulate a highly detailed version of a cortical hypercolumn.

The second line of thought tries to overcome the inherent limitations of software simulations by designing so-called neuromorphic hardware systems (see e.g. *Mead* [1989]; *Indiveri et al.* [2006]; *Merolla and Boahen* [2006]; *Renaud et al.* [2007]; *Vogelstein et al.* [2007]; *Indiveri* [2008]). Software simulations suffer from two major drawbacks. First of all, a lot of resources are spent on computing the dynamics of the simulated units, i.e. numerically integrating the corresponding differential equations. Secondly, the architecture of the processors ultimately running the program is inherently serial, so a lot of computing cycles are needed to update the state of every single neuron in the network, which in biological reality of course happens all at the same time. Both these difficulties can be tackled by designing electronic circuits which behave similarly to the components of a biological neural network. Such a *silicon neural network* does not need to integrate any differential equation, since its inherent dynamics are (nearly) identical to their biological counterparts. These hardware circuits can work and exchange signals independently of one another, so there is no need for time consuming serial processing because the implemented solution is inherently parallel. This neuromorphic hardware approach has a highly welcome *side-effect*: the small size of the constituent components (on the order of micrometers or less) implies very short time constants for the ongoing electrodynamic processes. This results in a speedup factor of several orders of magnitude with respect to biological timescales.

The FACETS[1] research project (*FACETS* [2009]) unites a highly interdisciplinary team of researchers in its quest for understanding the computational principles and paradigms behind the human brain. Biologists try to measure characteristics of single cells and larger structures within the brain. Mathematicians and physicists try to model neural networks and examine their behavior with computational and analytical methods. A number of workgroups, including the Electronic Vision(s) group in Heidelberg, are involved in building a large scale neuromorphic hardware device as a substrate for emulating neural network models. This so-called wafer-scale system is, within its physical limitations, freely configurable to emulate nearly arbitrary neural networks with an acceleration factor of up to 5 orders of magnitude with respect to biological real time (*Schemmel et al.* [2007, 2008]). On the FACETS wafer-scale neuromorphic hardware, up to 200.000 neurons and 50 million synapses can be emulated in parallel. This system is designed to serve as a universal tool for neuroscientific modeling, as the neuron and synapse circuits in the hardware reflect the behavior of state-of-the art neuron and synapse models.

[1] Fast Analog Computing with Emerging Transient States

1.3 The FACETS Demonstrator Project

Such an immensely complex system with such an enormous configuration space requires an equivalently complex software framework for its maintenance and operation. Since hardware design an software engineering are, to a large extent, complementary lines of work, it makes sense to have them done in parallel. However, for the testing of the developed software or of the operation workflow, a test bench in the form of a functional model of the yet unfinished hardware system is indispensable. For this purpose, an abstract high-level description of the given device was developed, with interfaces that resemble the ones implemented in the physical hardware.

The FACETS wafer-scale neuromorphic hardware system is composed of sophisticated subunits for which every detail is, in principle, known. Nevertheless, because of its huge complexity, it is impossible to predict the behavior of the system as a whole until actually operating it. Especially the impact of the system's intrinsic properties onto the dynamics of emulated neural networks can not be analytically estimated. With the real wafer-scale system not yet being available, only an *executable system specification* with behavioral models of all components can expose these consequences. By investigating the behavior of this *virtual hardware*, one can provide the scientific community with instruction and information about the capability and the limitations of the system already in early stages of development, long before it is physically available. Giving researchers a feeling for the huge potential of this novel neuroscientific tool is essential to pave the way for the success of the FACETS Hardware.

Within FACETS, a sub-project called the *FACETS Demonstrator* was initiated, which aims to demonstrate the successful interdisciplinary approach of the project. The capability of the neuromorphic hardware to perform brain-like computation shall be tested and demonstrated with biologically relevant cortical network models. For this purpose, neuroscientists involved in FACETS provided a versatile set of cortical *benchmark models* for emulation on the FACETS wafer-scale system. As long as the real FACETS Hardware is not available, the executable system specification has to take its place. This virtual hardware also has the advantage of not having any imperfections due to the semiconductor production process, which is very convenient, as it also serves as a reliable test bench for the operating software.

1.4 Outline

One aim of this thesis is the enhancement of an existing system simulation of the FACETS wafer-scale system. This executable system specification is then used as a test bench for the software layers designed to operate the FACETS Hardware. Furthermore, three FACETS Demonstrator benchmark models are simulated with the enhanced virtual hardware.

The thesis starts with a description of the FACETS wafer-scale system, its operating software and its existing virtual version (the system simulation) in chapter 2. In chapter 3 the three chosen FACETS Demonstrator benchmarks are presented. As a next step, the modifications made to the existing system simulation will be described in chapter 4, with a particular focus on the analog circuits responsible for the emulation of neuron and synapse dynamics. Furthermore, the developed methods and a software module for the transformation of parameters from the biological domain into the corresponding hardware configuration is presented in chapter 5. In the next chapter (6), the modified system simulation and with this also the developed parameter transformation is tested. Last but not least, the neural network bench-

mark models for the FACETS Hardware are emulated with the virtual hardware and their performance is analyzed in order to illustrate and quantify the capabilities and constraints of the FACETS wafer-scale system.

2 FACETS Neuromorphic Computing Framework

In this chapter the reader is provided with an overview of the FACETS wafer-scale neuromorphic hardware system as well as the software layers that are needed to operate such a system. Furthermore, the executable system specification of the wafer-scale system is presented. The level of detail varies according to the relevance for this thesis.

2.1 Wafer-Scale Hardware System

(Most of the information about the FACETS wafer-scale hardware system was taken out of Schemmel et al. [2008] and Fieres et al. [2008] or acquired by personal communication with members of the Electronic Vision(s) group.)
The FACETS Stage 2[1] Hardware system basically consists of a very large number of ASICs[2] containing the circuits emulating neurons and synapses. The desire to arbitrarily interconnect a large number of neurons leads to the wafer-scale neuromorphic system presented here. The technology used is a 180nm CMOS[3] process. The chip-manufacturing starts with a so-called *wafer* of monocrystalline silicon, to which a series of treatments is applied in order to build integrated circuits into the semiconductor material. A typical wafer is $1mm$ thick and has a circular geometry with a diameter ranging from $150mm$ to $450mm$. Due to technical limitations[4] the maximum size of a chip is limited to a defined fraction of the wafer, the so-called *reticle*. In the case of the FACETS wafer-scale system the size of the reticle is $20mm \times 20mm$. Each reticle contains eight analog neural network chips called HICANN[5]. The HICANN chip is the primary building block of this system and hosts up to 512 Neurons and 130k synapses, it is described in section 2.1.1. The reticle is applied onto an 8-inch (200 mm) wafer such that the HICANN chip is replicated around 400 times. HICANNs are not only connected with their neighbors within one reticle but also with the those belonging to adjacent reticles. This is possible through a method called wafer-scale integration: a post-processing step applies additional wires onto the wafer, such that reticles are interconnected, which makes the required high connection density possible. In order to utilize the neuromorphic substrate, it has to be connected to the outside world. This is realized through a stack of digital communication units ending at a host computer (see section 2.1.3). In addition to this

[1] The term *Stage 2* corresponds to the FACETS *wafer-scale* neuromorphic hardware, *Stage 1* corresponds to the *single chip system* based on the neuromorphic chip *Spikey*, which has been developed earlier at the Electronic Vision(s) group in Heidelberg
[2] Application Specific Integrated Circuit
[3] Complementary Metal Oxide Semiconductor
[4] During chip-manufacturing photolithography is used to mark areas in the silicon, which shall be later treated in a certain way, e.g. doped. A pattern is applied with a laser beam that passes a photomask, the size of which is limited for VLSI (very-large-scale integration).
[5] High Input Count Analog Neural Network

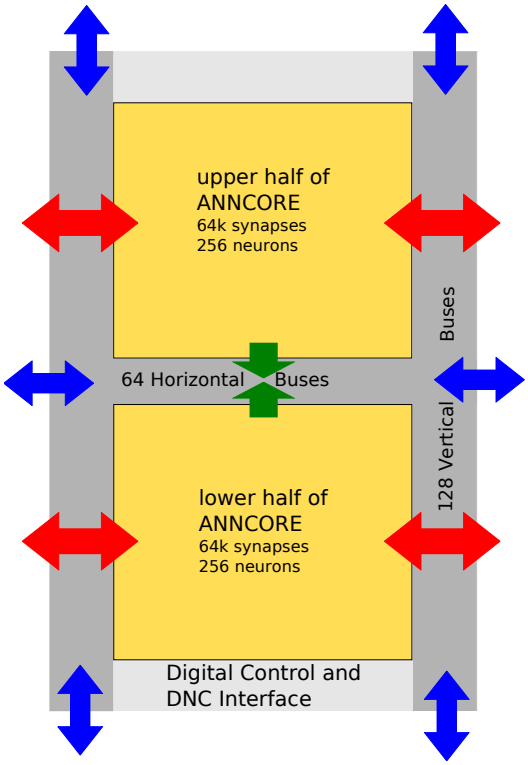


Figure 2.1: HICANN block structure with connectivity structure of the Layer 1 network, which is responsible for the neuron-to-neuron communication. For a description see the text in section 2.1.1.

the stack of digital communication can interconnect several wafer-scale systems, which allows to emulate neural networks that are too large to fit onto one wafer.

2.1.1 The HICANN Chip

The HICANN chip is the primary building block of the FACETS wafer-scale system, it contains all analog circuits emulating neuron and synapse behavior as well as the interface to the host via the DNC (cf. 2.1.3). In figure 2.1, a block diagram of the HICANN chip is shown. The biggest part is occupied by the Analog Neural Network Core (ANNCORE), which contains the synapse and neuron circuits. The ANNCORE will be described in a separate section (2.1.2) and for now is regarded just as the source and target of neural pulses. Remaining functional units of the HICANN chip are the so-called *Layer 1 network*, which is responsible for the transport of spike events to ANNCOREs on the whole wafer, and the *digital part* of the HICANN that controls the rest of the chip and communicates with the outside world.

2.1.1.1 On-Wafer Neuron-to-Neuron Communication (Layer 1)

The communication of spikes is performed with a mixture of time division and space division multiplexing. The network for the pulse event transportation consists of 64 horizontal buses running through the middle of the chip and 128 vertical buses on each side of the ANNCORE, as it is depicted in figure 2.1. One bus carries the spikes from up to 64 neurons by transmitting a 6-bit digital signal, that encodes the currently sending neuron (with an ID from 0 to 63). The ANNCORE generates these digital signals and feeds them into one of up to eight horizontal buses (green arrows). A signal of a horizontal bus can be switched to vertical buses in the left and right of the HICANN (not shown in the figure). Repeaters, which send or receive

Layer 1 events to or from adjacent HICANNs, are located at the HICANN borders(blue arrows). Thereby, the on-wafer communication network is able to connect every HICANN with any other on the wafer.
Signals on the vertical buses can be switched to one of 64 *synapse drivers* on each side of an ANNCORE block, where the signal is further processed. The signal can be also injected to the ANNCORE of the neighbor HICANN (red arrows).
The Layer 1 buses work asynchronously and reach an rate of up to 1.6GBit/s each.
Each HICANN has 8 circuits for the generation of pseudo-random spike sources implemented with a shift register. Each of them can send on one of the eight horizontal buses that are fed by the ANNCORE. The firing rate of each can be set individually.

2.1.1.2 Digital Units of the Chip

The remaining digital part of the chip has two major tasks:
The first is to communicate with the digital network chip (DNC), which is carried out by the *DNC Interface*. It exchanges pulse events as well as configuration and request packets with the DNC. While received pulse events are fed into the Layer 1 network, the configuration packets are decoded and processed by the *chip control* module. These configuration and control operations of the chip are the second major task.

2.1.2 Analog Neural Network Core (ANNCORE)

The task of the ANNCORE is mainly to receive a pulse from a pre-synaptic neuron from Layer 1, then translate the pulses into synaptic signals that end in the neuron circuits. In the neuron circuits the neuron behavior is emulated and possibly a spike is detected and a pulse is transmitted back into the Layer 1 and propagated to other cells. Figure 2.2 depicts the structure of the ANNCORE. The largest space is occupied by the synapse arrays containing 256×256 synapse circuits.[6] *Synapse Drivers* at the left and right of the synapse handle the pre-synaptic signals from the Layer 1 network and operate the synapses circuits. The short-term plasticity mechanism is included in the synapse driver circuits. There are 256 neuron circuits in each block of the ANNCORE, each of these so-called *denmems* can receive synaptic input from all synapses of the same column. 64 neurons share a priority encoder, which turns one neuron signal after another into a 6-bit Layer 1 signal. This signal is then transmitted to the Layer 1 network and to the DNC Interface. The ANNCORE contains also circuits for long-term synaptic plasticity, which are located above the upper and below the lower synapse block.

2.1.2.1 Synapse Drivers

The synapse drivers are the interface for the Layer 1 events to enter the ANNCORE. They are alternately located on the left and right of the synapse array, one synapse driver *drives* the synapses of two rows. Hence, there are 64 synapse drivers located on each side of the synapse block that can be fed with serial 6-bit signals from Layer 1 resulting in a maximum of 64 pre-synaptic inputs per driver. A synapse driver is connected to the synapses via one of 4 different strobe lines. At arrival, a serial pulse event it is deserialized and the 6-bit

[6] Note that the first produced version of the HICANN chip contains only 232×256 synapses per block, as 14 synapse rows could not be realized due to missing space on the chip. Nevertheless, in the whole course of this thesis the HICANN is assumed to hold 256 rows of synapses per block.

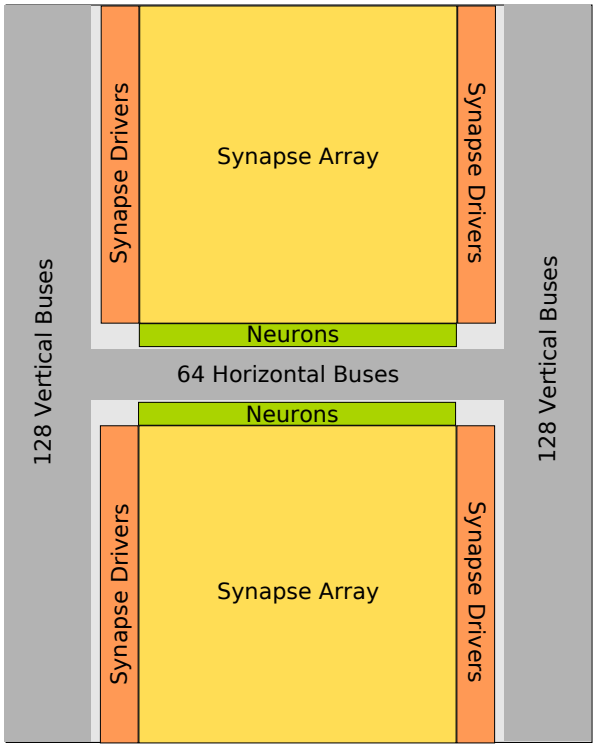


Figure 2.2: Schematic of the ANNCORE with Layer 1 buses. Elements not depicted in this figure: the *Priority Encoders*, which generate Layer 1 signals for the neuron-to-neuron communication, and the circuits implementing a *long-term plasticity* mechanism. For a description see the text in section 2.1.2.

sender address is split up into two parts. The upper two bits encode which of the 4 strobe lines a pulse is sent on to the synapse circuits. The length of these strobe pulses τ_{stdf} can be modulated by a short-term plasticity mechanism, which is described later in this paragraph. The lower 4 bits of the 6-bit neuron address are transmitted to the synapse array. In order to avoid signal attenuation on Layer 1 buses due to feeding one lane multiple times into one ANNCORE, the signal received by one synapse driver can be mirrored to an arbitrary number of adjacent drivers above and below.

2.1.2.2 Synapse Circuits

The synapse circuit contains a 4-bit address decoder and a 4-bit static RAM storing the synaptic weight. If the subsequently received 4 bits from the synapse driver match the ones stored in the address decoder, an output current is generated. The transmitted signal is a square current pulse with amplitude $weight \cdot g_{\text{max}}$ and length τ_{stdf}. The stored digital weight is translated by a digital-to-analog converter and determines the fraction of g_{max} to be applied as the amplitude of the square pulse. The maximum conductance g_{max} can only be set individually for every synapse row, and thus is responsible for the base value of the strength of synaptic connections. This results in a maximum of 16 different synaptic weights for 256 synapses belonging to one row. The output current is injected into one of two available inputs of the neuron circuit, where it modulates the course of a synaptic conductance. For every row it has to be decided in the synapse driver, whether the synapses connect to the first or second input of the *denmem*. In a typical setting the first input is used to emulate an excitatory synaptic conductance, the second one emulates an inhibitory synaptic conductance. This means that the synapses belonging to one row are either excitatory of inhibitory.

2.1.2.3 Short-Term Plasticity (STP)

Remark: Almost everything of the sections (2.1.2.3, 2.1.2.4, 2.1.2.5) was taken from the diploma-thesis of Johannes Bill, section I.1.3, as the described circuits is almost the same in the HICANN chip as in the protoype chip Spikey

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

2.1.2.4 Hardware Model

Every *synapse driver* of the FACETS Stage 2 Hardware supports two modes of *short-term plasticity*, namely *depression* and *facilitation*. The basic idea is to introduce a time varying *inactive partition* I with $0 \leq I \leq 1$. It decays exponentially with time constant t_{rec}, while every AP processed by the synapse driver increases I by a fixed fraction U_{SE} towards the maximum. This idea leads to the following dynamics for the inactive partition:

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

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

$$w_{\text{dep}} \propto 1 - \lambda \cdot I \tag{2.1}$$

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

$$w_{\text{fac}} \propto 1 + \lambda \cdot (I - N) \tag{2.2}$$

2.1.2.5 Hardware Implementation

The model presented above results in additional circuitry in the synapse driver (see fig. 2.3). This paragraph outlines the basic operation of the STP mechanism. For details see *Schemmel et al.* [2007].
Short-term synaptic plasticity can be switched on via the `enable`-signal. The `mode`-signal selects the type of STP. The inactive partition I corresponds to the voltage V_{I}.[7] The current I_{rec} adjusts the decay time constant t_{rec}. When triggered by a spike, charge is transferred from C_1 to the adjustable capacitor C_2[8]. This leads to a rapid change of V_{I}:

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

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

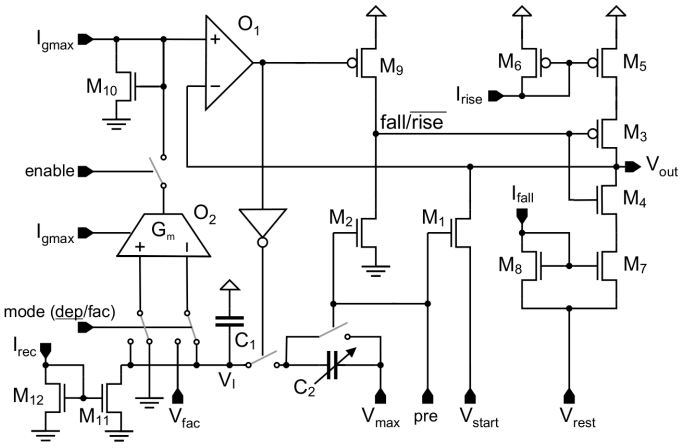


Figure 2.3: Synapse driver circuit implementing short-term plasticity. The STP mechanism is located in the lower left part of the circuit. This diagram actually shows the circuits of the FACETS Stage 1 Hardware. For the short-term plasticity there is just one major change in the HICANN chip: The capacitor C_2 is replicated 64 times, one for every pre-synaptic neuron feeding this synapse driver.

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

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

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

In the HICANN chip the capacitor C_2 is replicated 64 times, each one implements the inactive partition I for each of the 64 pre-synaptic neuron feeding this synapse driver.

A suitable configuration for STP is presented in Section 5.2 considering all constraints and trying to mimic the original model of short-term plasticity by *Markram et al.* [1998].

2.1.2.6 Long-Term Plasticity

In addition to STP, which changes the efficacy of synapses in the range of biological seconds, a mechanism implementing *long-term potentiation and depression* (time range: seconds to hours) is built into the chip. This *spike-time dependent plasticity* (STDP) may change the digital 4-bit weight of the synapses. In the first produced version of the HICANN chip this mechanism is disabled. As this mechanism is only of secondary importance for this thesis, the reader is referred to *Bi and Poo* [1997] and *Morrison et al.* [2008] for the biological data and models and to *Schemmel* [2006] for the hardware implementations.

2.1.2.7 Membrane Circuits

The hardware neuron circuits implement an adaptive exponential integrate-and-fire model according to *Brette and Gerstner* [2005] described by differential equations (2.3) with a variable speedup-factor of 10^3 and 10^5. One membrane circuit called *denmem* has two circuits for synaptic input. The output current from every of the 256 synapses belonging to the column above the denmem is injected into one of its input circuits, where it modulates the size of a conductance g between an adjustable so called *reversal potential* and the membrane capacitance. The two synaptic inputs can used for e.g. *excitatory* and *inhibitory* input. Up to 64 *denmems* can be grouped together to form a neuron by connecting its membrane capacitances, such that the maximum number of input synapses per neuron is $\approx 16K$ with the possibility to set different values for nearly every parameter of the denmem (e.g. multiple excitatory reversal potentials and time constants can be set for a single neuron)

$$\begin{aligned} C\frac{dV}{dt} &= -g_{\mathrm{L}}\,(V - E_{\mathrm{L}}) + g_{\mathrm{L}}\,\Delta_{\mathrm{T}}\ \ \exp(\frac{V - V_{\mathrm{T}}}{\Delta_{\mathrm{T}}}) - w + I \\ \tau_{\mathrm{w}}\frac{dw}{dt} &= a(V - E_{\mathrm{L}}) - w \end{aligned} \tag{2.3}$$

The first equation describes the dynamics of the neuron's membrane potential V. The voltage is coupled with the adaption current w. Synaptic input is injected through the current I. E_{l} denotes the leak reversal potential to which the voltage converges if no more input is applied. g_{L} is the leakage conductance that controls the speed of this convergence. The exponential term describes the process of an action potential (AP) generation. The threshold V_{T} and the slope factor Δ_{T} determine the start of the exponential nonlinearity. a denotes the adaption coupling parameter and τ_{w} the corresponding time constant. A spike is usually detected when V reaches a finite value V_{spike}. The downswing of V after an action potential is not described by the equations but introduced with a reset mechanism at the spike time t_{spike}:

$$\begin{aligned} V &\rightarrow V_{\mathrm{reset}} \\ w &\rightarrow w + b \end{aligned} \tag{2.4}$$

In biology current flows into the cell trough ion channels, which are activated by neurotransmitters that are released by synapses. In this neuron model ion channels are represented by a conductance between a synaptic reversal potential and the membrane voltage V.
Thus:

$$I = g_{\mathrm{exc}}(t)(V - E_{\mathrm{rev,e}}) + g_{\mathrm{inh}}(t)(V - E_{\mathrm{rev,i}}) \tag{2.5}$$

where $g_{\mathrm{exc/inh}}$ and $E_{\mathrm{rev,exc/inh}}$ denote excitatory and inhibitory conductances and reversal potentials. An incoming spike triggers an increase of the conductance by the weight *weight*

$$g \rightarrow g + weight \tag{2.6}$$

The synaptic conductance then decays with time constant τ:

$$\tau\frac{dg}{dt} = -g(t) \tag{2.7}$$

The ADEX neuron model is capable of reproducing all known biological firing patterns (see *Touboul and Brette* [2008]). Simulation of the hardware circuits implementing this model have shown, that nearly all of this firing patterns are also reproducible by the *denmems* (Personal communication from *M.-O. Schwartz and S. Millner*).

2.1.3 Stack of Digital Communication Units (Layer 2)

The HICANN chip emulates neuron dynamics with a speedup factor of about 10^4 compared to biological neurons. Assuming 512 neurons per chip, each firing with a mean biological frequency of $10Hz$, more than $50 \times 10^6 spikes/s$ are generated in one HICANN. For an appropriate analysis of network dynamics one needs to record most of these spikes. This requires a high bandwidth for the communication of these spikes to a host computer, therefore a hierarchical stack of hardware units was developed, that configures and interacts with the wafer. This communication is also referred to as *Layer 2* communication. 8 HICANNs are linked to one so-called DNC[9] over a *2-Gigabit* connection each. 4 DNCs communicate with one FPGA[10] over a *16 Gigabit* connection. DNCs and FPGAs are mounted on a printable circuit board (PCB). The connection setup of these units is shown is figure 2.4. A total of 12 FPGAs

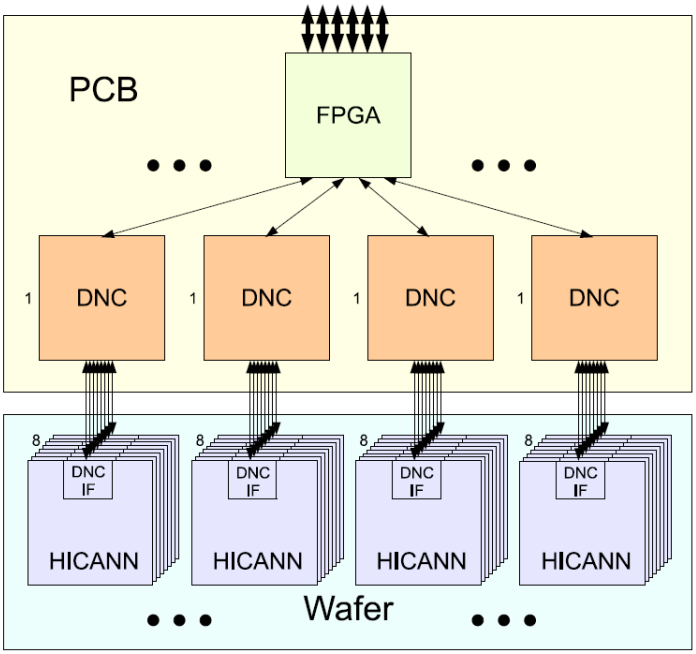


Figure 2.4: Wafer-scale system setup with component connections. FPGAs and DNCs are located on the printable circuit board (PCB). 1 FPGA is connected to 4 DNCs, each of which has a link to 8 HICANNs on the wafer.

are needed to operate one fully configured FACETS wafer-scale system. FPGAs receive data from a host computer which is mainly divided into configuration and pulse packets. Configuration packets are processed and possibly further transmitted to the addressed sub-units, where the same procedure is repeated. The Layer 2 is not only responsible for the recording, but also for the transport of spikes *to* and *from* HICANN chips on *different* wafers. Therefore FPGAs have to be interconnected. For the correct on-time delivery of spikes the DNCs and FPGAs implement a special routing. Every neural event carries a target and a time at which it shall be delivered to a HICANN. As the time that is needed for the transmission of such a digital event package is not a priori predictable and depends on the overall network activity,

[9] Digital Network Chip
[10] Field Programmable Gate Array

these pulses are delayed and then injected into the Layer 1 network at the correct point in time. This feature offers the possibility to realize synapses with an *adjustable delay*. This is contrary to the on-wafer neuron-to-neuron communication, where pulses are transported nearly instantaneously.
A schematic of the future wafer-scale system is shown in figure 2.5 containing also the mechanical and electrical supplementary infrastructure.

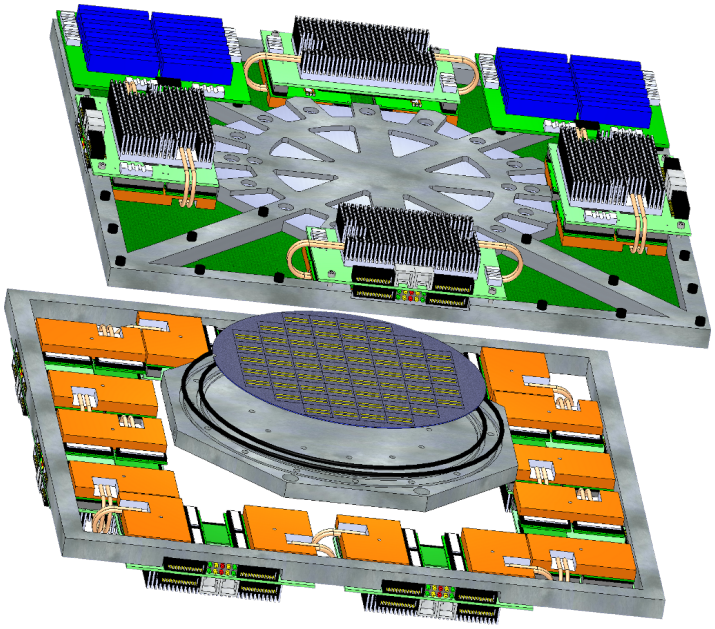


Figure 2.5: Schematic of the FACETS wafer-scale system. In the center the wafer with the reticle structure is shown. Above and below there are the printable circuit boards containing FPGAs and Digital Network Chips. Further the mechanical (aluminium frames and plate) and electrical (blue, at the top) support modules are illustrated.

2.2 Software Layers for the Operation of the Wafer-Scale System

A system like the FACETS Stage 2 Hardware, as described in section 2.1, which is very complex and highly configurable, needs an adequate software counterpart. Biological neural network models have to be mapped onto the hardware system, the configuration has to be specified and later converted into a protocol that is processable by the hardware. The FACETS neuromorphic hardware is designed to serve as flexible modeling tool for neuroscientists. Therefore an user-friendly interface is indispensable, which conceals the complexity of the system but at the same time provides all the advantages to the user.

The software stack developed for this task is depicted in figure 2.6. The highest level is *PyNN*, a high level modeling tool for neural networks. A biological network architecture is first described by a directed hypergraph (*BioGraph*), which is then translated into its counterpart of the neuromorphic hardware system, the so-called *HardwareGraph*. This highly complex process is conducted by the so-called *MappingTool*. The *Stage 2 Configurator* finally undertakes the configuration of the hardware system, runs the simulation and returns the

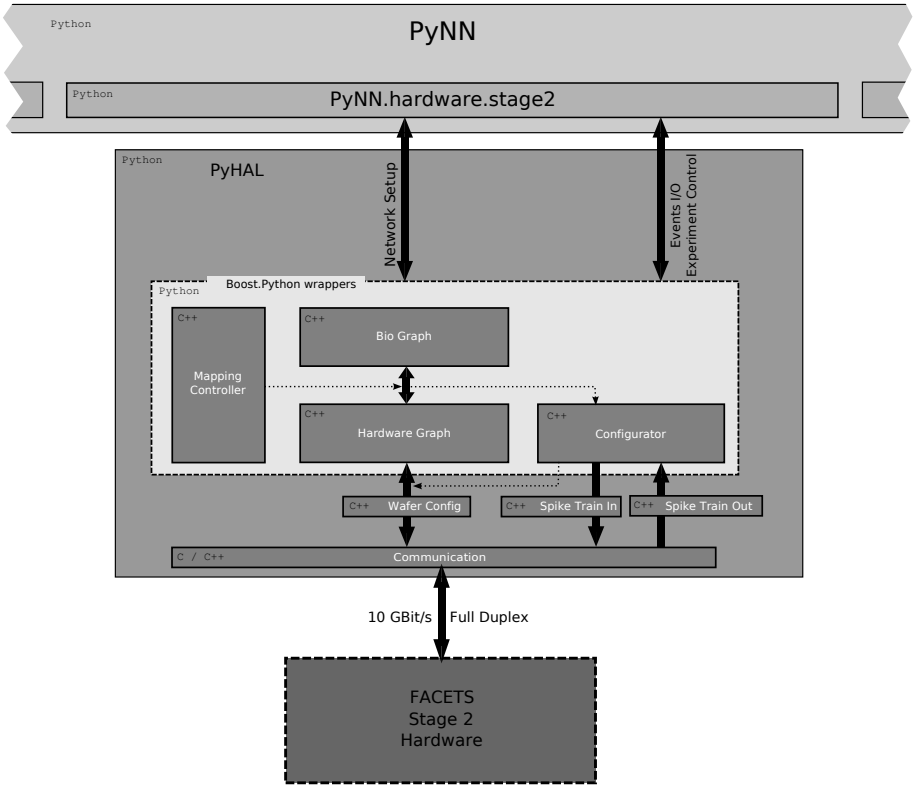


Figure 2.6: Software layer controlling the FACETS Stage 2 system. The top level is the meta-language PyNN, in which neural network models are described. A given network model is then converted into a graph representation by the PyNN interface to the FACETS Hardware (PyNN.hardware.stage2). The MappingTool translates this network into a hardware configuration. Finally the FACETS is configured and operated by the Configurator. Figure taken from *Brüderle* [2009].

retrieved data to the user via PyNN.

2.2.1 PyNN

A large number of simulation tools for neural networks have come up over time, each with its specific focus and own user interface. For the comparison of different simulation back-ends and for the cross-check of results a unified language for the description of neural network models has been developed within the FACETS project: With PyNN[11] (*Davison et al.* [2008]) network models can be built once and the same simulation can be run later on any back-end supported by PyNN, currently *NEST*(*Gewaltig and Diesmann* [2007]), *NEURON*(*Hines et al.* [2008]), *Brian*(*Brette and Goodman* [2008]), *PCSIM*(*Pecevski and Natschläger* [2008]) and the FACETS Hardware.
PyNN distinguishes between a low-level and a high-level API[12]: A number of standard types of cells and synapses are provided. Individual cells can be either `create`d and `connect`ed, or groups (`Populations`) of the same type can be defined. Between these populations or cells connection patterns can be applied (so-called `Projections`).

The FACETS Hardware can be operated from PyNN. The PyNN interface to the FACETS wafer-scale system `PyNN.hardware.stage2` handles the instructions of a given network model and operates the lower software layers described in the following paragraphs. Having the same interface for the neuromorphic hardware and established software simulators allows to run and compare neural network simulations by just switching one line in the code, the one that describes the simulator to use (supposed that both backends support the used cell and synapse types):

```
import pyNN.nest as simulator → import pyNN.hardware.stage2
```

The PyNN.hardware.stage2 module also supports the executable system specification of the FACETS wafer-scale system (see section 2.3 and chapter 4), such that we can run neural network experiments described in the meta-language PyNN on the virtual version of the FACETS wafer-scale system and compare the results with simulations performed with software simulators. *Experiments* of this kind are presented in chapter 6.

2.2.2 GraphModel

The core of the operating software stack is the so-called *FACETS GraphModel* developed by the project partner at the TU Dresden (*Wendt et al.* [2008]), which holds all information of both the biological network model and the FACETS hardware. The PyNN.hardware.stage2 module creates a graph of the biological neural network described in the meta-language PyNN. In this *BioGraph* the whole topology and setup of the neural network is stored. The FACETS GraphModel consists of nodes, hierarchical and directed named edges, as well as hyperedges. E.g. neurons are nodes within the graph, synapses are realized through directed edges between two neurons and every node or edge may have a connection to its parameter node.
The *HardwareGraph* represents the structure and the configuration of the FACETS hardware system. It contains all hardware modules in a hierarchical structure (e.g. FPGAs, DNCs and HICANNs). If units are connected in the hardware, e.g. a horizontal Layer 1 bus with a vertical one, the same connection is drawn in the HardwareGraph with a directed named

[11] Python Neural Networks
[12] Application Programming Interface

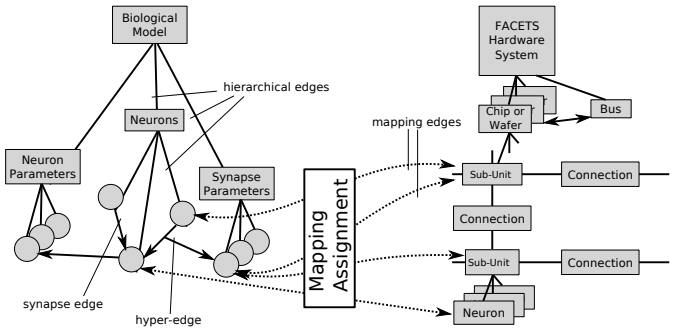


Figure 2.7: FACETS GraphModel. One can see the BioGraph at the left and the HardwareGraph at the right. Mapping assignments of biological elements to its hardware counterparts are depicted. Figure taken from *Brüderle* [2009].

edge.
With the so-called *mapping edges* the BioGraph can be linked to the HardwareGraph, e.g. a biological neuron is *mapped* to its representation in the HardwareGraph, a membrane circuit. An exemplary graph is depicted in figure 2.7.

2.2.3 MappingTool

The translation of a biological neural network into a system like the FACETS hardware is quite complex, as resources are limited or parameters can not be set to arbitrary values with the desired precision. The MappingTool, mainly developed at the TU Dresden, aims at optimizing this procedure, such that the hardware representation resembles the biological network as precisely as possible. For that a series of algorithms have to be applied, starting with the *placement* of biological neurons onto the wafer, going on with drawing synaptic connections on and off the wafer (Layer 1 and Layer 2 *routing*), ending with the *transformation* of parameters. This is a task of multi-objective optimization, as several shortcomings and limitations have to be considered. And even one procedure affects the outcome of the others.

2.2.3.1 Neuron Placement

There are many requirements for an accurate placement of neurons onto the wafer:
One of the most limited resources of the FACETS wafer are Layer 1 buses for synaptic connections (see section 2.1.1.1 and *Fieres et al.* [2008]). In large networks some synaptic connections can *not* be realized due to the limited number of horizontal and vertical buses. This synapse loss may have serious impact on the mapped neural network model. The utilization of these buses can be reduced, if neurons are placed in an intelligent manner. This can be done by e.g. placing neurons with similar outgoing and / or incoming connections onto the same HICANN. Furthermore, neurons have to be grouped according to their type (excitatory or inhibitory): Hardware neurons that are connected to the same priority encoder send on the same Layer 1 bus(cf. section 2.1.2). Thus their spikes enter the ANNCORE through just one synapse driver, where the type of the synaptic connection is set.

The FACETS project partners from TU Dresden developed a force-based clustering algorithm that is able to cluster biological neurons having similar properties (*Wendt et al.* [2007]). This *NForceCluster* algorithm operates in an abstract multi-dimensional space where neurons are arranged according to their distance in terms of properties. Neurons are then clustered into groups that fit onto a HICANN and are finally mapped to the corresponding hardware circuits in the HardwareGraph. This leads to a much higher mapping efficiency compared to a random placement, i.e. a lower loss of synapses and/or reduced space needed on the wafer. At the current state, neurons can be arranged according to the following properties: common incoming synapses, common outgoing synapses and the neuron type (excitatory or inhibitory). The weighting of each property can be set individually.

2.2.3.2 Routing of Synaptic Connections

After the placement the neurons have to be connected (*routed*). For inter-wafer connections the FPGAs and DNCs have to be configured accordingly. For intra-wafer connections the Layer 1 buses have to be switched, such that every desired pulse reaches its target HICANN. Furthermore, the address decoders in the ANNCORE have to be set. For that task a sophisticated *routing* algorithm was developed in the Electronic Vision(s) group in Heidelberg, that aims to realize as much synaptic connections as possible. At the same time this algorithm is able to prioritize special synapses to make sure, that these connections are realized in the network. For a detailed description of this routing algorithm see *Fieres et al.* [2008].
After the placing and routing of the network, it can be evaluated whether all neurons and synapses of the biological network have been realized on the neuromorphic substrate. The *MappingTool* offers the possibility to provide a PyNN-description of the distorted network. This model can later be run on a software simulator to analyze e.g. the influence of an occurred synapse loss.

2.2.3.3 Parameter Transformation

As a last step neuron and synapse parameters have to be transformed into adequate electrical parameter values. This transformation is challenged by lots of difficulties: parameters can just be defined with limited precision, single biological parameters are represented by multiple hardware parameters, not every hardware parameter can be set individually for every biological value but is shared for a group of functional units, the neuromorphic hardware works at quite a different time-scale. This parameter transformation is part of the author's work for this thesis and is described in detail in chapter 5.

2.2.4 Configuration and Control of the FACETS Hardware

The lowest layer in the stack needed for the operation of the FACETS hardware is the so-called *Stage 2 Configurator*. This module reads out the configuration data from the GraphModel and builds data packets that can be processed by the hardware system. In advance to that, the connection to the FPGAs of the system is established, to which the data is transmitted. After the successful configuration of the FACETS hardware spike trains are sent to it as stimuli for the hardware neurons. This triggers the emulation of neuroscientific experiments on the FACETS hardware. The Configurator receives and collects pulse data from the hardware and also may interact with the running system (e.g. readout or change of synaptic weights). In a future version this shall be also possible from PyNN in an *interactive mode*. After running the

simulation the Configurator delivers the acquired results back to the PyNN module, where it is further handled.

2.3 Executable System Specification of the FACETS Wafer-Scale Hardware System

Within the FACETS project and as a substantial part of the work for this thesis, an executable system simulation model of the final FACETS hardware platform has been developed to characterize the routing architecture, to identify communication bottlenecks and to examine the impact of limited and discretized parameter value ranges. This simulation model not only serves as a constant feedback for the developers of the hardware architecture design but also as a specification for the operating software described in section 2.2. In this section an overview is given about the state of the system simulation, with which the author was faced at the beginning of his work. Later on, a short introduction into the system modeling language *SystemC* is provided, as it will be the major tool for the further development of the executable system specification in chapter 4.

2.3.1 Initial State: Overview

The system simulation is a high-level description of the FACETS Stage 2 hardware system, and was realized in the simulation environment of *NCSim* [13]. NCSim is capable of running various simulation tools within one framework. At the moment it supports *VHDL*[14] and *VerilogAMS*, which are hardware description languages for digital, mixed-signal and analog (only supported by VerilogAMS) system design, and *SystemC*, a system modeling language based on C++, which is described more precisely in the next section. Thus, different modules of a large system, described in differing modeling languages, can be simulated and verified together in one run. This, for example, provides the possibility to simulate the behavior of one module very precisely while using very abstract versions of other components.

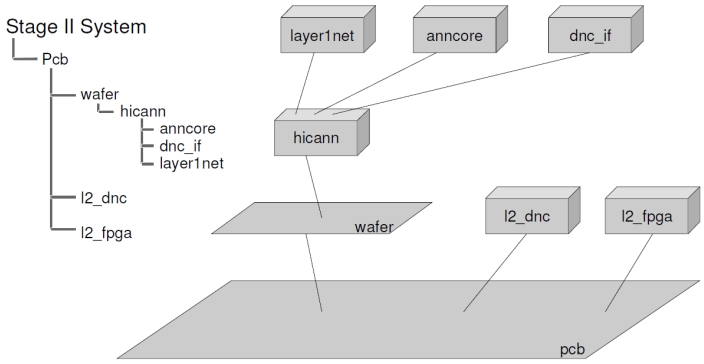


Figure 2.8: Hierarchical Structure of the executable system specification of the FACETS wafer-scale Hardware.

[13] *NCSim* stands for *Incisive Simulator* and is a unified simulation engine for the design and verification of digital and analog systems (e.g. ASICs or FPGAs) from *Cadence Design Systems*

[14] Very High Speed Integrated Hardware Description Language

In Figure 2.8 the hierarchical structure of the virtual FACETS wafer-scale System is depicted. The top-level module instantiates one or more *printable circuit boards* **pcb**, which are the containers for the *FPGAs* (**l2__fpga**), *Digital Network Chips* (**l2__dnc**), both responsible for the digital-event communication (cf. section 2.1.3, and the wafer containing the **hicann** modules with its sub-units (cf. section 2.1.1).

The simulated modules are connected to one another according to the real FACETS Stage 2 system and provide the same functionality. The behavioral SystemC models of the FPGAs and DNCs implement all the routing mechanisms for the digital event communication and are limited by the same buffer sizes and maximum bandwidths as the real system. A schematic of the HICANN SystemC module is shown and described in figure 2.9, reflecting the state of development at the beginning of the work for this thesis. While all other modules are simulated in SystemC or C++, the Layer 1 module **layer1net** is described and simulated in *VerilogAMS*, which means that every signal on every wire has to be computed at simulation time. Obviously, this massively slows down the run time of a simulation. The ANNCORE at that state already implements correct pulse routing from synapse drivers to the assigned neurons. However, the implemented neurons do not mimic any behavior of real neurons, they just fire at every incoming spike and no timing-dependent features are integrated, such as e.g. a short-term synaptic plasticity mechanism. The model as a whole can be configured and operated via configuration and pulse files or via packets received by the FPGA. The FPGA then translates this data into Layer 2 communication packets which are further propagated to the DNCs and HICANNs, where the correct configuration and creation of pulse events on the Layer 1 is performed.

At this state the system simulation is already fully executable, it provides the full functionality for routing of spikes over Layer 1 and Layer 2. However, it lacks the speed of execution that is necessary for the application of e.g. the cortical benchmark models described in 3 and misses a correct behavioral model of the ANNCORE.

2.3.2 Introduction to the System Modeling Language SystemC

SystemC is a modeling language for system design based on C++ made for the system level of design (for a detailed information see *Groetker et al.* [2002]). With its event-driven simulation kernel SystemC provides the possibility to run processes concurrently, as it is often needed when designing highly parallel working systems. One can arbitrarily choose the abstraction level of design for the whole simulation (e.g. timing accuracy) and separately for every sub-unit.

In a *transaction-level model* (TLM) the communication between different modules is performed via function calls. A transaction-level model is accurate in terms of functionality and sometimes in terms of timing but usually it is not accurate in terms of structure.

In a *pin-accurate, cycle-accurate hardware model* modules are accurate at their boundaries in these two characteristics. In addition, the modules are functionally accurate, but the internal structure is not necessarily the same as in the real implementation.

A model at the *register transfer level* (RTL) is pin- and cycle-accurate at its boundaries, but also its internal structure reflects the registers and combinational logic of the target implementation.

SystemC allows to simulate models of different levels of accuracy in one environment. For example, a TLM module can communicate with a RTL module by switching a transactor between these modules. Thereby, one can test one module at a very detailed implementation

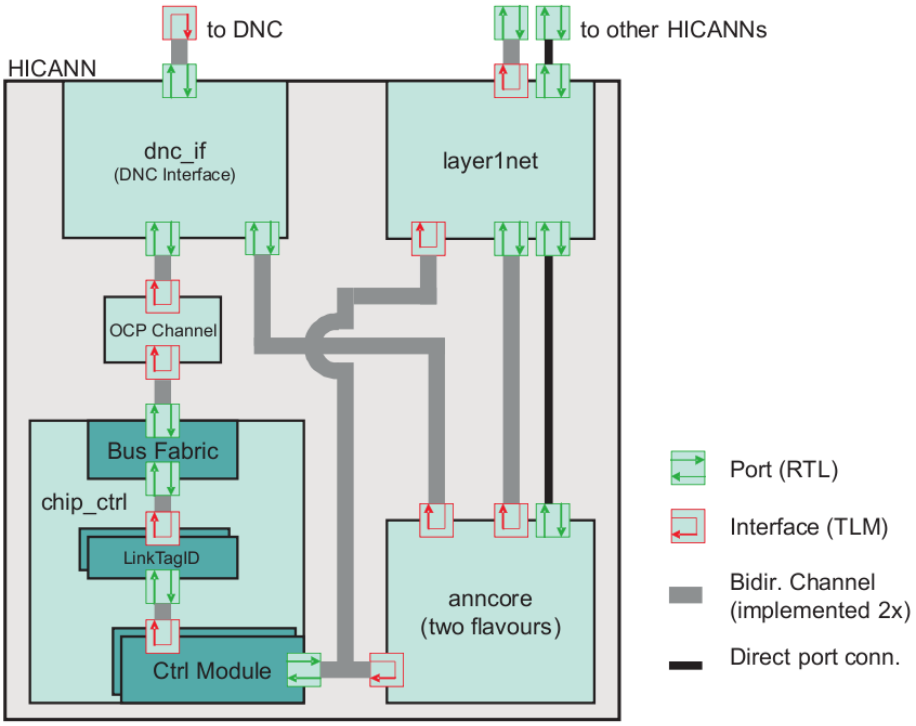


Figure 2.9: Schematic of the HICANN in SystemC (Effective January 2009), reflecting the state of development before the start of the work for this thesis. In the upper left the DNC Interface **dnc_if** is depicted. It receives pulse events and configuration data from the Digital Network Chip. While pulse events are directly fed into the Layer 1 network, configuration data is communicated to the **chip control** module in the lower left, where the task of configuration of the **anncore** and **layer1net** is undertaken. Signals of the Layer 1 network are communicated via ports and are implemented in *VerilogAMS*, all other modules in *SystemC*. For an explanation of ports and interfaces see section 2.3.2

level (e.g. at the RTL) with an abstract test bench module, or, one can iteratively refine a system from a model that is pin-accurate at the beginning to a model at the RTL at the end. Furthermore, with SystemC one can create an *executable specification* of a model, which can e.g. serve as a test bench for the software that will operate and communicate with this design.

The basic SystemC core language offers the following elements:

- `Modules` are the basic building blocks storing data types and implementing algorithms
- `Ports` are the gates for modules to connect and communicate with their surroundings
- `Interfaces` provide a set of operations just specified by their name, passed parameters and return values. An interface is implemented by a module or a channel, that is derived from interface such that there can be different implementations for one interface. [15]
- `Channels` build the communication units, which carry out the operations defined in an interface. SystemC provides a number of elementary communication mechanisms from simple wires and buffers to FIFOs. A modeler can design his or her own channels in SystemC.
- `Processes` are the basic functional units. In the SystemC simulation kernel they can be executed concurrently in opposition to typical programming languages where processes are executed sequentially.
- `Events` trigger or resume the execution of processes representing a condition that may occur during simulation. An event object can be notified by a process or a channel to trigger all its associated processes immediately, after a delta-delay or after time t.

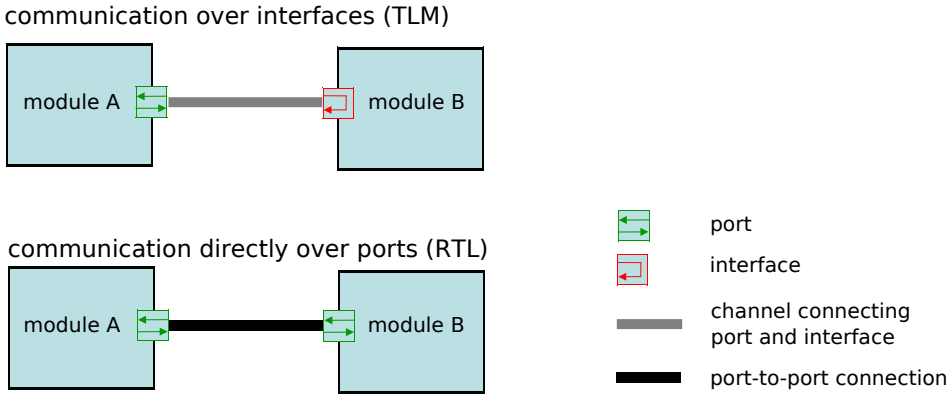


Figure 2.10: Example of different models for communication in SystemC. In the top modules A and B communicate over a port-to-interface connection (i.e. via function calls). At the bottom the modules are directly connected from port to port communicating with primitive signals. This method is used for pin-accurate modeling while the first one is used for transaction-level modeling.

In a *transaction-level* model communication between two modules is performed in the following way, see also figure 2.10: Module A has an outgoing *port* that is connected to an *interface* in module B. If A wants to communicate something with B, it just calls one of the functions defined in the interface. Also in a *pin-accurate* model a module has an outgoing port

[15] In Object Oriented Programming an interface is a purely virtual base class

but this time, it is connected to an incoming port in module B. The connection is realized through a primitive *signal*. For the communication the value of the signal has to be changed, for example, if this connection implements just a single digital signal, the *signal* connecting the two ports can be switched from 1 to 0.
Timing is brought into effect through the interplay of *processes* and *events*. For example, process A wants to trigger process B with a delay of 10 seconds. Process B is sensitive to an *event* called *trigger_process_b*, this means that every time when this event gets *activated*, process B is executed. Thus, process A can notify the event *trigger_process_b* to get active in 10 seconds. This way process B will be executed after 10 seconds.

With these basic features and the possibility to build own modules, channels, ports and interfaces, SystemC becomes a very flexible and powerful tool for system modeling.

3 FACETS Demonstrator Benchmark Models

The neural networks presented in this section shall serve as benchmarks for the FACETS wafer-scale system. In order to prove the functionality and universality of the FACETS hardware (2.1) and its operation software framework (cf. section 2.2) as a neuroscientific modeling tool, the cortical models must be of biological relevance and represent a large variety of network dynamics and paradigms. All models are described in *PyNN* (cf. section 2.2.1), so that they can be both *simulated* with neural software simulators and *emulated* with the FACETS hardware. A high scalability of the models is required to test the influence of hardware constraints (e.g. loss of synapses when large networks are mapped onto the wafer).

3.1 KTH Layer 2/3 Attractor Memory

FACETS project partners from the KTH Stockholm provided the community with an *attractor memory model of the neocortical layer 2/3*. The network architecture is based on *Lundqvist et al.* [2006].

The human neocortex consists of 6 distinct layers. Each of them is specified by its nerve cell types and the destination of their afferent and efferent[1] connections. A columnar structure of the cortex was discovered by *Mountcastle* [1979]: When moving an electrode perpendicular to the surface of the cortex, it encounters cells, which are activated by the same sensory input. When moving the electrode parallel to the cortex surface, one notices sharp changes in the attenuation of cortical neurons at every $200 - 300\mu$m. Therefore, it was conjectured that about $10^4 - 10^5$ neurons are organized in an approximately cylindrical so-called vertical *hypercolumn*, with a diameter of $0.2 - 0.3$mm. The vertical connections in this area are much denser than horizontal ones. Neurons belonging to one hypercolumn have nearly identical receptive fields[2]. One hypercolumn holds between 50 and 100 *minicolumns* with about 80 neurons each. It was proposed that minicolumns - *not* single neurons - are the *smallest computational units* in the cortex (*Lundqvist et al.* [2006]). Each of minicolumn may react to a certain feature of the hypercolumn's receptive field, which enables a hypercolumn to process many different features of its receptive field.

The Layer 2/3 Attractor Memory model is the result of applying an associative attractor memory architecture to the columnar structure of the cortex. The model uses neuron and synapse models that correspond to biological measurements. *In vitro* experiments support the attractor memory paradigm, as local UP states[3] of about 0.5% of pyramidal cells in the cortical layer 2/3 were observed. These neurons simultaneously enter in a state of high activity

[1] afferent transmit signals into a cell and efferent out of the cell

[2] The *receptive field* of a neuron in a *high* cortical region is the set of all neurons in a *lower* region, from which it receives its afferent connections. A *lower* cortical region is considered to be closer to the associated receptor cell population.

[3] Neurons have two preferred subthreshold membrane potentials. These states are referred to as UP and DOWN states. In the experiment a cell in an UP state increases its membrane potential by some 10mV, also its spiking frequency increases to 20Hz.

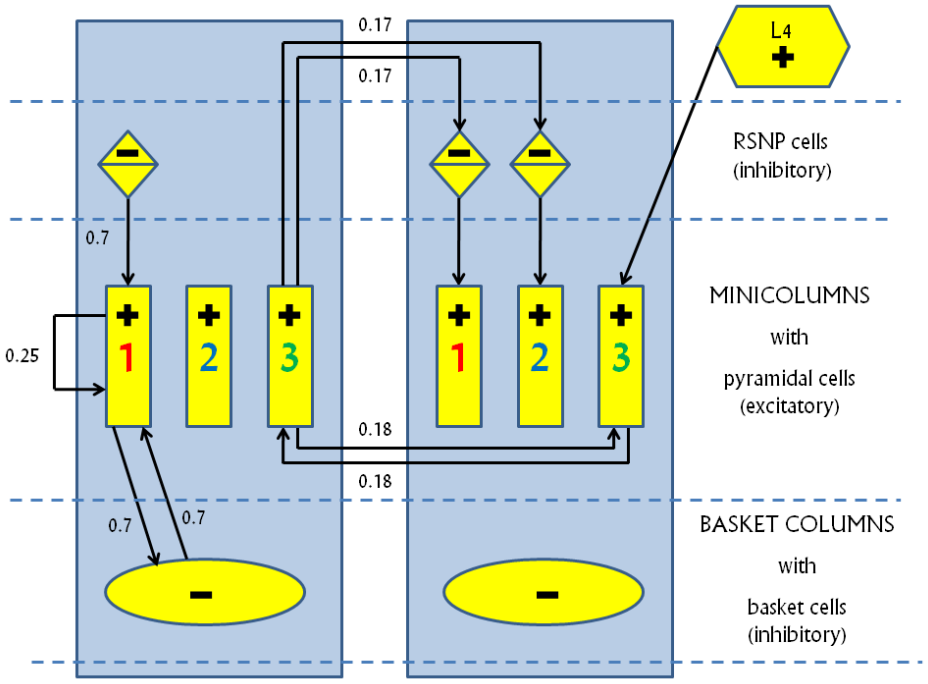


Figure 3.1: Schematic of the Layer 2/3 attractor memory model. Description in the text, section 3.1.1. This schematic was provided by *Mihai Petrovici*

and an increased mean membrane voltage that lasts for several hundreds of milliseconds. One possible explanation for these states are attractor dynamics (see *Cossart et al.* [2003] and *Shu et al.* [2003]). This model is able to perform critical cell assembly operations, such as pattern recognition, completion and rivalry (*Lundqvist et al.* [2006]).

3.1.1 Network Model

The Layer 2/3 model chosen as a FACETS Benchmark has an elaborated structure which is shown in figure 3.1. The model contains a number of hypercolumns depicted in blue. Minicolumns (yellow rectangles) in a hypercolumn compete in a winner-take-all fashion for a state of high activity. A minicolumn contains 30 interconnected excitatory pyramidal cells, each of which receives an individual excitatory input from a Poisson-type spike source in Layer 4. Each minicolumn receives inhibitory input from 2 RSNP[4] cells depicted as yellow diamonds. The overall activity of all minicolumns in a hypercolumn is damped through a negative feedback provided by inhibitory interneurons in the basket column (basket cell populations are depicted as yellow ellipses). Every attractor is represented by one minicolumn per hypercolumn. Minicolumns belonging to an attractor form a so-called *patterns* (there are three patterns in figure 3.1). They excite each other via long-range connections, thus enabling pattern completion: even when a single minicolumn receives input from Layer 4, it spreads its activity to all other minicolumns belonging to the same pattern. Minicolumns belonging to different attractors and hypercolumns inhibit each each other via RSNP cells, thus creating a long-range winner-take-all mechanism leading to a pattern rivalry.

[4]Regular spiking non pyramidal

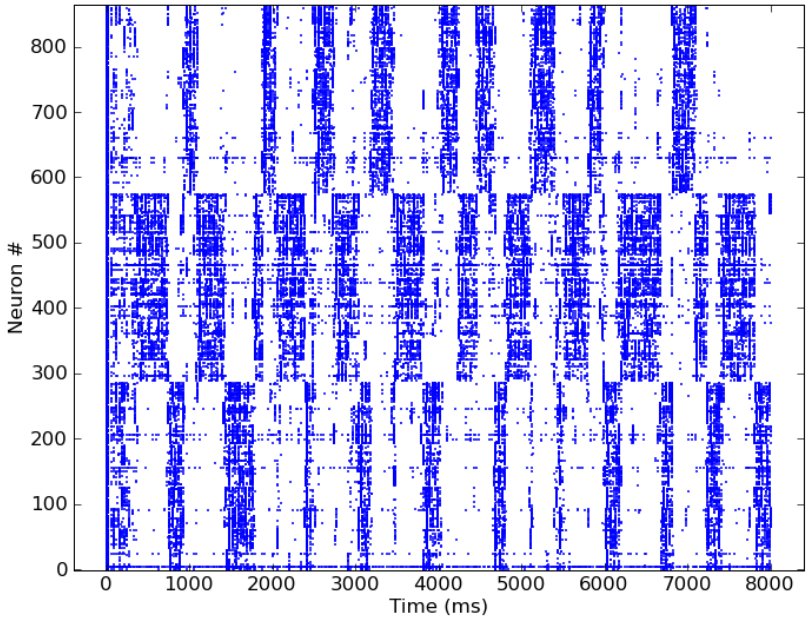


Figure 3.2: Spiking patterns of the Layer 2/3 Attractor Network simulated with NEST shown in a raster plot. Each dot denotes a spike of a certain neuron at time t. The spike trains of 9 hypercolumns with 3 minicolumns each are plotted. Minicolumns belonging to the same attractor are grouped together. The activity of the basket and RSNP cells is not plotted.

The *default* benchmark configuration consists of 9 hypercolumns with 3 minicolumns each, with a total of 936 neurons. However, the model is fully scalable in both the number hyper- and minicolumns. The connection probabilities are given in figure 3.1. As only a fraction of a biologically realistic number of minicolumns are implemented, the strength of synaptic connections has been scaled up such that every neuron receives the same input as measured in experiments.

3.1.2 Example

Exemplary spiking patterns of this model can be seen in figure 3.2. This example network contains 9 hypercolumns with 3 minicolumns each, thus giving rise to 3 different attractors. Minicolumns belonging to the same pattern are grouped together. One can see that all the neurons of one attractor simultaneously enter in a state of high firing rate, which lasts for several hundred milliseconds. Each pyramidal cell in the minicolumns receives a Poisson-type input of 4000Hz. For the RSNP and pyramidal cells, an adapting neuron model is used. All synapses (except the ones connecting the Poisson input) exhibit short-term plasticity dynamics (excitatory synapses are depressing, inhibitory synapses are facilitating).

3.1.3 Measure of Stability

The most basic qualitative measure for this model is whether it exhibits any attractor patterns at all. One quantitative measure is the average attractor dwell time, i.e. the average duration

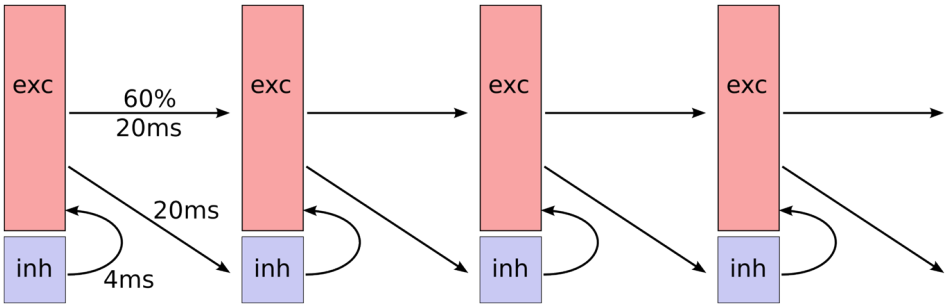


Figure 3.3: Schematic of the Synfire Chain with feed-forward inhibition

of the attractor state. An attractor state can be recognized by an increased *mean firing rate* and an increase of the *mean membrane voltage* of the cells belonging to the active pattern. Analysis has pointed out (*Petrovici et al.* [to be published 2010] that the latter condition is a much more reliable measure than the former. Another qualitative measure is to check whether the model is able to perform pattern recognition. For this, the input from Layer 4 has to be changed, such that the different attractors respond to different input patterns.

3.2 Synfire Chain with Feed-Forward Inhibition

A *synfire chain* (*Abeles* [1982]) is a feed-forward network that consists of a chain of neuron groups, each projecting to the following group. Each neuron has many excitatory connections to neurons of the next group. This means that neurons of one group receive a highly correlated input.
Such a network setup allows a stable propagation of synchronous spikes in cortical neural networks (*Diesmann et al.* [1999]). The model presented here is based on *Kremkow et al.* [submitted 2009]. It has been developed by FACETS participants from ALUF[5] and INCM[6] and adds a feed-forward inhibition mechanism, which limits the number of spikes generated within an excited neuron.

3.2.1 Network Model

The network architecture of this *synfire chain* is depicted in figure 3.3. One *synfire group* consists of 80% excitatory and 20% inhibitory neurons. These numbers have been chosen in order to resemble the neuron distribution in the neocortex. The two sub-groups have no recurrent connections. Each neuron of the inhibitory sub-group is connected to 65% of the excitatory neurons of the same group with a synaptic delay of 4ms. Excitatory neurons stimulate 65% of both excitatory and inhibitory neurons of the following group with a delay of 20ms. The first synfire group is stimulated as if it had a preceding group, in which every neuron fires once (the exact firing times are Gauss-distributed with a given σ). The reaction of the network to such a pulse packet can be seen in figure 3.4: in the left panel one can recognize a stable synfire chain, when feed-forward inhibition is enabled. In the right panel the response of the network is shown when feed-forward inhibition is disabled, i.e. the inhibitory cells are removed. Not only does the width of the pulse packet diverge, but also the number of times

[5] Albert-Ludwigs-Universität Freiburg
[6] Institut de Neurosciences Cognitives de la Méditerranée, Marseille

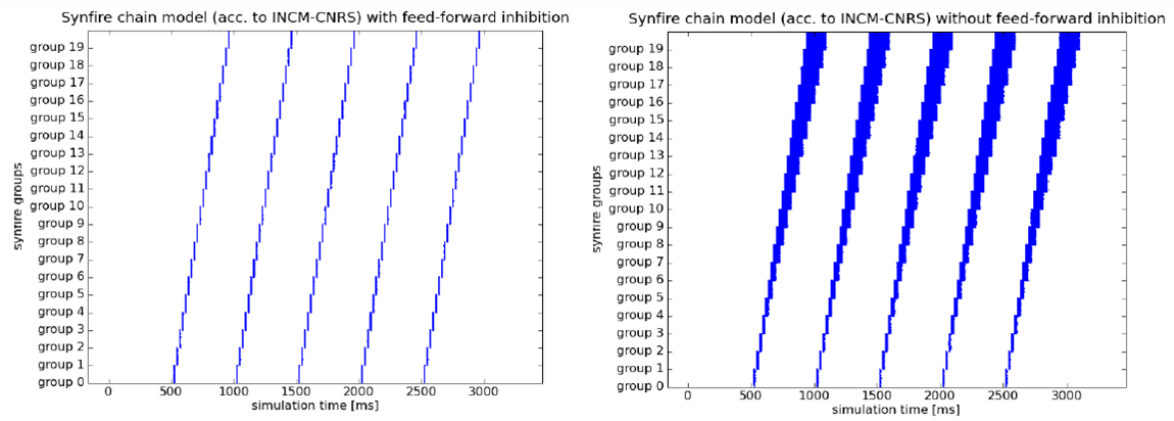


Figure 3.4: Rasterplots of a stimulated Synfire Chain with and without feed-forward inhibition simulated with NEST. On the left feed-forward inhibition is enabled, such that a *clean* synfire attractor is achieved. On the right feed-forward inhibition is disabled, the pulse packets diverges

an excited neuron fires. The pulse packet diverges without the throttling inhibitory group. The network can be easily scaled up by either increasing the number of neurons per group or the number of groups.

3.2.2 Measure of Stability

The presented synfire chain with feed-forward inhibition is self-stabilizing in the manner that the width of the pulse packets stay approximately constant in later synfire groups. In such a *synfire attractor* every excitatory neuron of a synfire group should spike exactly once within a narrow time window. In order to compare implementations of this model on the FACETS Wafer-Scale System with software simulations, one should first compare the *total number of spikes per group* and later the *jittering of the spikes*, that means the standard deviation of the spike times. The closer the total number of spikes to the number of neurons per group and the lower the mean deviation of the spike times, the better the functionality of the synfire chain.

3.3 Self-Sustaining Cortical Activity with Asynchronous Irregular Firing Patterns

The third FACETS Demonstrator benchmark is a model of self sustained activity with asynchronous irregular firing patterns in the cortex, which is based on *Destexhe* [2009] and was provided by FACETS project partners from UNIC[7], in Gif-sur-Yvette.

The activity of single cortical neurons in awake animals is quite noisy, with irregular discharges at frequencies between 1 and 20Hz (*Matsumura et al.* [1988]). Simulated networks composed of integrate-and-fire neurons can reproduce this activity pattern, showing *asynchronous irregular*(AI) states. In order to display characteristics similar to in-vivo measurements large networks of a few thousand integrate-and-fire (IF) neurons are needed (*Vogels and Abbott* [2005], *Boustani et al.* [2007], *Kumar et al.* [2008]). The model chosen tries to be

[7]Integrative and Computational Neuroscience Unit

as close as possible to the biological reality, it does not use simple IF but *adaptive exponential* integrate-and-fire neurons (ADEX). This neuron model reflects the highly complex behavior of biological neurons including phenomena such as adaptation, bursting, or the ability for an inhibitory rebound (see section 2.1.2.7 and *Brette and Gerstner* [2005], *Touboul and Brette* [2008]).

3.3.1 Network Model

The model consists of 80% excitatory pyramidal cells and 20% inhibitory interneurons which are randomly connected with a probability of 2%. Excitatory cells are regular-spiking, while inhibitory are fast-spiking[8]. The standard network according to *Destexhe* [2009] consists of 500 neurons. 10% of all cells are initially stimulated for 50ms, and depending on the parameters the activity is self-sustained or dies out. With this setup the generation of AI states is possible, this means that single cells spike irregularly and that the pairwise correlation of the spike trains of two neurons is low, i.e. they don not fire in synchrony. The network gets much more stable if 5% of the excitatory cells are replaced by low-threshold spiking (LTS) neurons. LTS neurons are capable of a rebound burst, i.e. the neuron bursts in response to an input when it is in a hyperpolarized state (the membrane voltage is below the resting potential).
When scaling up the network size, the connectivity has to remain constant, while synaptic weights have to be reduced such that every neuron still receives the same total input.

3.3.2 Quantification of the Network States

The following two properties can be used to quantify and describe the network dynamics: regularity and synchrony.
The regularity can be estimated by computing the coefficient of variation (CV) of the inter-spike intervals (ISI), averaged over all network cells:

$$CV_{\mathrm{ISI}} = \left\langle \frac{\sigma_{\mathrm{i}}^{\mathrm{ISI}}}{ISI_{\mathrm{i}}} \right\rangle \tag{3.1}$$

$\mathrm{ISI_i}$ and $\sigma_{\mathrm{i}}^{\mathrm{ISI}}$ denote the mean and the standard deviation of the inter-spike intervals of a neuron, the brackets $\langle\rangle$ indicate averaging over different cells. The $\mathrm{CV_{ISI}}$ takes on large values ≥ 1 for temporally irregular systems. (The CV for a Poisson processes is 1)

The synchrony can be quantified by calculating the average cross-correlation (CC) between arbitrary pairs of neurons in the network:

$$CC = \left\langle \frac{Cov(S_{\mathrm{i}}, S_{\mathrm{j}})}{\sigma(S_{\mathrm{i}})\sigma(S_{\mathrm{j}})} \right\rangle \tag{3.2}$$

$\mathrm{Cov}(S_{\mathrm{i}}, S_{\mathrm{j}})$ denotes the covariance between two spike counts S_{i} and S_{j}, and $\sigma(S_{\mathrm{i,j}})$ is the standard deviation of each spike count. The average $\langle\rangle$ should be taken over at least 500 pairs. The network is considered to be in an asynchronous state if CC is low enough (typically < 0.1).

[8] *Regular spiking* neurons respond with a constant firing rate to a constant current input. Fast-spiking neurons behave the same but with a higher firing rate

4 The Executable System Specification - New Strategy and Components

As motivated in the introduction of this thesis (see 1.3) a further development of the executable system specification becomes indispensable, if:

1. The virtual hardware is supposed to serve as a realistic test bench for all layers of the operation software presented in Section 2.2.

2. The capability of the system to serve as universal tool for neuroscientific modeling shall be demonstrated with the models from Section 3.

3. Thereby, the constraints and consequences of the general design shall be determined.

In this chapter, the reader is provided with the changes made to the system simulation compared to its state introduced in Section 2.3. The focus is on the implementation of the behavioral model of the Analog Neural Network Core conducted by the author. At the end an overview is given about the current state of the full virtual FACETS Stage 2 Hardware and the simplifications compared to the real system are listed.

4.1 Changes Applied to the Wafer and HICANN Level

One of the most urgent needs for the refinement of the *virtual hardware*[1] is a speed up in simulation duration. Thus, all modules implemented on the Register-Transfer-Level (RTL) should be replaced by Transaction Level Modeling (TLM) solutions. The only module simulated at the *register transfer level* was the *VerilogAMS* implementation of the *Layer 1 network*, which is pin-accurate and cycle-accurate. This means that every single wire on the real chip has its representation in the module and is connected time-accurately to other modules where the signals are translated into abstract function calls. So the *wire* Layer 1 network was removed on the level of a whole wafer and replaced by a behavioral model of Layer 1 written in SystemC - one for each HICANN. The behavioral Layer 1 modules interact with one another over interfaces, that means via function calls and not via digital signals on pins.
The actual implementation, configuration and testing of the behavioral Layer 1 module was successfully conducted in parallel tho the author's work by *Bernhard Kaplan*.

On the HICANN level the following changes were applied by the author (see also figure 4.1: All modules responsible for the digital control of the chip except the **dnc_if** were replaced by one appropriate TLM module. This means, that the configuration of all submodules is no more performed via configuration packets, which are generated by the host and transmitted via Layer 2 communication to the HICANN DNC Interface.
Instead, the configuration data is retrieved by direct access to the Hardware GraphModel(see

[1] the terms *virtual hardware*, *executable system specification* and *system simulation* are use synonymously in this thesis

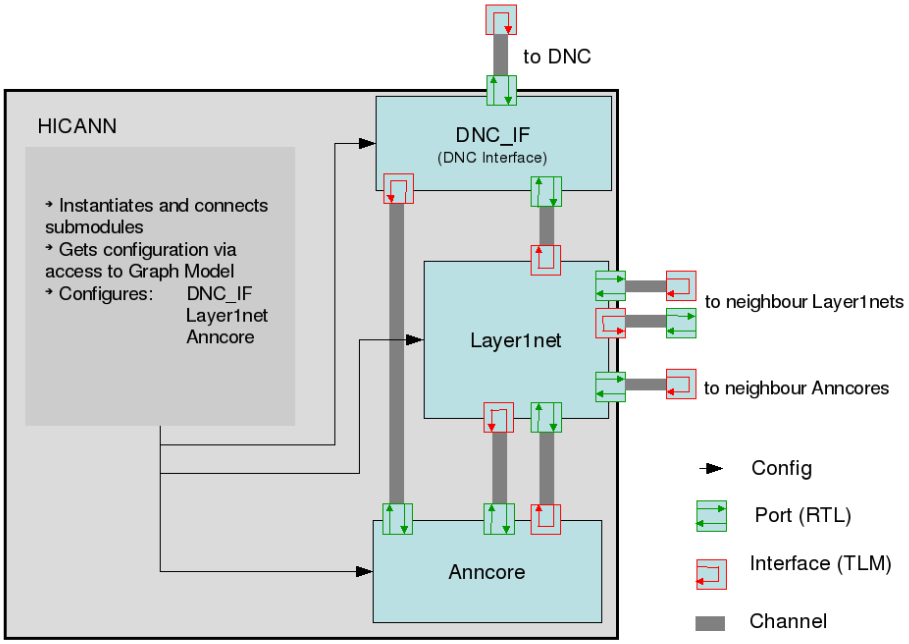


Figure 4.1: Schematic of the HICANN in SystemC. This module is basically just the container for its submodules: the interface to the Digital Network Chip (**DNC_IF**), the Layer 1 network (**Layer1net**), and the **Anncore**. The **DNC_IF** receives pulse data from the DNC and injects it into the **Layer1net**. The Layer1net module connects to other Layer1net modules of adjacent HICANNs and to the ANNCORE of the same container. This way all neural events can be easily propagated and distributed to all ANNCOREs on one Wafer. In the ANNCORE these spikes are further processed and possibly a response is transmitted back into the Layer 1 network and to the DNC Interface, where it can be recorded and transfered to higher modules of the digital event communication. In the current state of the executable system simulation, the configuration of the HICANN's submodules is not performed via packages received from the DNC, but via direct access to the Hardware GraphModel.

section 2.2.3). Technically, before the start of the simulation every HICANN is assigned a pointer to a C++ class called *gm_access*, which was developed by the project partners of the TU Dresden and allows a user-friendly interface to all needed data stored in the GraphModel. With this acquired data the HICANN carries out the configuration of the **Anncore**, **Layer1net** and **DNC_IF**.
This new procedure of configuration embodies a high abstraction from the real hardware model, but was necessary due to the following reasons:

1. There were changes in the protocol of the configuration packages, so that the earlier implemented modules of the digital control of the chip were outdated.

2. No automated creation of the configuration packages with data acquired from the GraphModel is available.

3. The new procedure benefits from a speedup in simulation time, which is one of the main targets of the revised system simulation.

This solution is essential for achieving an executable system simulation that reflects the full capability of the future wafer-scale system. Furthermore, the ANNCORE itself was substantially re-engineered by the author, as will be shown in the following section.

4.2 The Behavioral Model of the ANNCORE

The behavioral model of the ANNCORE contains all important functional units of its real counterpart. In Figure 4.2 a schematic of the *SystemC ANNCORE* is shown. Only the synapse drivers have their individual counterparts in the behavioral ANNCORE. From the synapse circuits two arrays with the digital synaptic weights and address decoder bits remain. Membrane circuits that are grouped together to emulate a single biological neuron with an increased input count are represented by one single behavioral neuron. The information which synapse column connects to which logical neuron is stored in a designated map (not shown in the figure).

Neural pulse events from the Layer 1 network enter the ANNCORE through one of the synapse drivers. Every pulse event carries the 6-bit ID of one of up to 64 pre-synaptic neurons sending on this bus. The received pulse event can be mirrored to the adjacent synapse drivers above or below. The SystemC synapse driver compares the 6-bit ID with the IDs stored in the synapse address decoder array and, in case of a match, inject a spike into the corresponding neuron instance. When a neuron fires, a Layer 1 event with a 6-bit neuron ID is generated and sent to both the behavioral model of the Layer 1 network and the DNC Interface of this HICANN.

4.2.1 Synapse Drivers and Synapse Array

The behavioral model of synapse driver can be configured directly with nearly all original hardware specific parameters. One can set the maximum conductance $g_{\mathrm{max,row}}$ and the synapse type (excitatory or inhibitory) for each of its two synapse rows. Furthermore a synapse driver keeps all hardware specific parameters concerning short-term plasticity. In the behavioral model the address decoding of Layer 1 events is simplified such that the 6-bit address is not split up into two parts as conducted in the real hardware. Instead, the synapse address

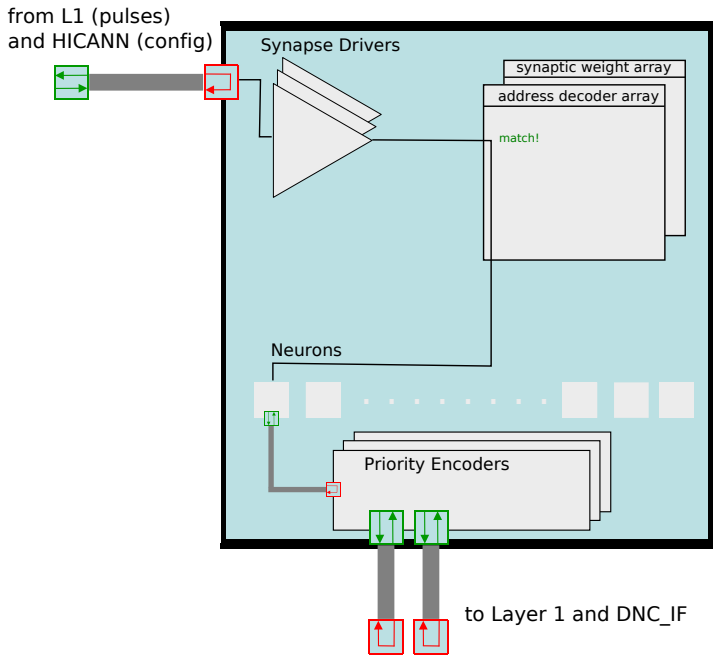


Figure 4.2: Schematic of the behavioral model of the ANNCORE with all basic functional units: synapse drivers (256), neurons (8 - 512), the synapse arrays (with 131K synapses) and the priority encoders. The module can be accessed via the simulated HICANN for configuration or from the virtual Layer 1 network for for sending pulse events (which contain the ID of the pre-synaptic neuron sending on the Layer 1 bus and the ID of the synapse driver to which it is addressed). When the latter happens, the pulse is sent to the corresponding synapse driver and possibly mirrored to adjacent drivers. The synapse driver then checks the two associated rows in the address decoder arrays. In case of a match the stored weight is read and possibly modulated by the short-term plasticity mechanism in the synapse driver. Then the evaluated synaptic response is sent to the neuron assigned to the active synapse column. If a spike is detected in the neuron module, it is further propagated to one of eight Priority Encoders, from which a pulse is fed back into the Layer 1 network or sent to the DNC Interface.

decoder array stores values between 0 and 63. (In contrast to the real hardware, where one synapse stores 4 bit and the upper two bits of of a received signal are decoded in the synapse driver.) The 4-bit digital weights w_{digital} are stored in a second array as integers from 0 to 15. Hence, for every incoming spike the synapse driver compares the received neuron ID with all addresses stored in the corresponding rows of the synapse address decoder array. In the case of matching the related digital weight is read from the weight array. Then the amplitude of a post-synaptic signal is generated with weight $w = g_{\text{max,row}} \cdot \frac{w_{\text{digital}}}{16}$. This information is sent to the designated neuron, where the course of the synaptic conductance is finally calculated.

4.2.2 Short-Term Synaptic Plasticity

The virtual hardware exactly captures the short-term plasticity mechanism implemented in the hardware, as described in Section 2.1.2.3.
Thus, in *depression* mode, the synaptic weight affecting a neuron is modulated in the following way, according to (2.1):

$$w_{\text{dep}} = w_{\text{max}} \cdot (1 - \lambda \cdot I) \tag{4.1}$$

Remember that I denotes the *inactive partition* of synaptic resources and $\lambda = \mu \cdot V_{\text{max}}$ the scaling factor, determined by hardware parameters.
In case of *facilitation* the synaptic weight applied to a neuron's conductance is given according to (2.2):

$$w_{\text{fac}} = w_{\text{max}} \cdot (1 + \lambda \cdot (I - N)) \tag{4.2}$$

Note that for the *facilitation* mode the factor $N = \frac{V_{\text{fac}}}{V_{\text{max}}}$ is added compared to *depression*, which ensures a wider range of facilitation.

For every pre-synaptic neuron, a tuple containing the time of the last spike t_{last} and the corresponding value of the inactive partition $I(t_{\text{last}})$ is stored. For every incoming spike the inactive partition I is updated:

$$I(t) = I(t_{\text{last}}) \cdot \exp\left(-\frac{t - t_{\text{last}}}{\tau_{\text{rec}}}\right)$$

Then, the pulse is sent to the neurons according to equations (2.1) or (2.2). In a last step, the inactive partition I is increased by the fixed fraction U_{SE} :

$$I = I + U_{\text{SE}} \cdot (1 - I)$$

and the current time is stored in t_{last}.

After the configuration of a synapse driver has been carried out with hardware specific parameters (e.g. V_{max}, V_{fac}, I_{rec} and the size of the Capacitor C_2), these parameters are used to calculate the abstract parameters used in the STP mechanism above.

4.2.3 Hardware Neurons

For the simulation of the ANNCORE to be as realistic as possible, an accurate model of the neuron circuits is essential. One approach would be to translate the dynamics of the real hardware neurons back into a functional model, as it was conducted with the STP mechanism in the section above. The presented solution is a different one. As the *denmem* circuit represents an implementation of the *adaptive exponential integrate and fire model* (ADEX) from *Brette and Gerstner* [2005], just this model is used in the virtual hardware. The ADEX

model is given by the differential equations (2.3) and has already been implemented in several established neural software simulators, such as *NEURON*, *NEST* or *BRIAN*. Thus, a possible strategy to include this model into the behavioral ANNCORE could have been to run a NEST instance for each neuron of one HICANN. However, in this approach one would have to interact with this running program at every incoming or outgoing spike, which would result in a significant communication overhead, along with a correspondingly longer simulation time. Therefore it was decided to design our own ADEX neuron model and include it into the existing executable system simulation. Another key reason for that decision was the following:
The software simulators addressed above are all time-driven, in contrast to the system simulation, which provides an event-driven simulation kernel.[2] Include a time-driven model into an event-driven environment would imply a significant performance loss, which is why it was regarded as mandatory to implement an event-based version of the ADEX neuron model and embed it into the system simulation.

An event-based neuron model needs the following functions to interface with its simulation environment.

- A function that updates a neuron after an *incoming spike* (e.g. calculate all variables of the neuron for the actual point in time and increase a synaptic conductance according to synapse's strength).
- A function that updates a neuron after an *outgoing spike* (e.g. update all variables according to the actual point in time and then reset the potential to the reset potential)
- A function that returns the *time of the next spike* under the assumption that there is no future input to this neuron

Hence, the procedure of an event-driven simulation is the following:

1. Determine the neuron that is going to spike next in time.
2. Update the state of this neuron (see update function above) and propagate the spike to all neurons, to which it has outgoing synaptic connections, and update also their state.
3. Re-calculate the time of the next spike for every neuron that is involved in this procedure.
4. Continue with the first step.

Fortunately, the event-driven simulation kernel provided by *SystemC* is able to perform the first task. The propagation of spikes is performed through the Layer 1 and Layer 2 communication. Thus, one just has to care about three function listed above.
It is also useful to provide another function:

- A function that provides a *refractory mechanism*,[3] which ensures that there is minimal time between two spikes. This function clamps the membrane voltage to the reset

[2] Time-driven means that the states of all neurons are updated in parallel after every time step *dt*. In an event-driven simulator the neuron variables are only updated, when a spike is received or emitted.

[3] In biology an action potential is not a delta peak, but has a certain width in time. During an AP no other depolarization can occur as Na+ Channels are closed. This time is denoted *absolute refractory time*. A *relative refractory mechanism* follows the first one: the hyperpolarization of the membrane potential inhibits the membrane to generate another AP.

potential but allows incoming events to update the neuron's variables, e.g. synaptic conductances.

4.2.3.1 Analytical Integrate-and-Fire Model with Synaptic Conductances

By changing (or removing) certain parameters or variables, on can reduce the rather complex ADEX neuron (the one implemented in the HICANN chip) to a much simpler model, for example, by removing the adaptation current and setting Δ_T to 0 one obtains a leaky integrate-and-fire neuron. For this neuron model, there exists an analytical solution (*Brette* [2006]), which means that spike times can be calculated exactly up to machine precision.

$$\begin{aligned} C\frac{dV}{dt} &= -g_L(V - E_\mathrm{L}) + g_\mathrm{exc}(V - E_\mathrm{rev,e}) + g_\mathrm{inh}(V - E_\mathrm{rev,i}) \\ \tau\frac{dg}{dt} &= -g \end{aligned} \tag{4.3}$$

Brette not only provides the mathematical concept for the exact calculation of this model but also the functions implementing this concept written in *C*. The model was included into the executable system simulation and for the first time it was possible run a simulation on the virtual hardware that showed exactly the same spiking behavior of a single neuron as the equivalent software simulation with *NEST*. However, this model not only misses an adaptation and an active spike generation mechanism but also has the major constraint that excitatory and inhibitory synaptic time constants have to be identical.

4.2.3.2 Event-based Adaptive Exponential Integrate-and-Fire Model

The ADEX neuron model implemented in the FACETS Stage 2 system is described by the system of two differential equations given in (2.3), which means that there is no way to calculate spike times other than using an approximation method (e.g. Runge-Kutta or simple Euler). Usually, these methods are used within time-driven simulations, where the states of all neurons are updated in parallel for every time step, with the consequence that a neural event occurred during one time bin cannot effect its surroundings before the next time step. Working in an event-driven simulation environment, it is not suitable to update every neuron at every time step, because this would result in a very long queue of events for the scheduler [4]. The management of this queue is assumed to be the major bottleneck of event-driven neural simulations, cf. *Brette et al.* [2006].

Therefore, task was to turn this analytically unsolvable model into an event-based neuron model, which changes the neuron variables only when a neural event (i.e. a spike) is received or released. Of course, one wants to predict the firing time of a neuron with as little computational cost as possible. Considering that the executable system simulation of the FACETS Stage 2 system is not supposed to serve as yet another neural software simulator, the author provides a simple event-based model of the implemented neuron circuits, which is not optimized in speed but easy to implement and also capable of recording the membrane voltage.

[4] The scheduler in an event-driven simulator sorts all tasks-to-be-done according to the time when the tasks shall be processed. It then starts to work off the job that comes first in the queue and then proceeds downwards, along the queue. During a simulation, the temporal priority of some tasks changes all the time. Thus, the queue containing all future tasks has to be rearranged very often, which is very expensive in terms of computational power.

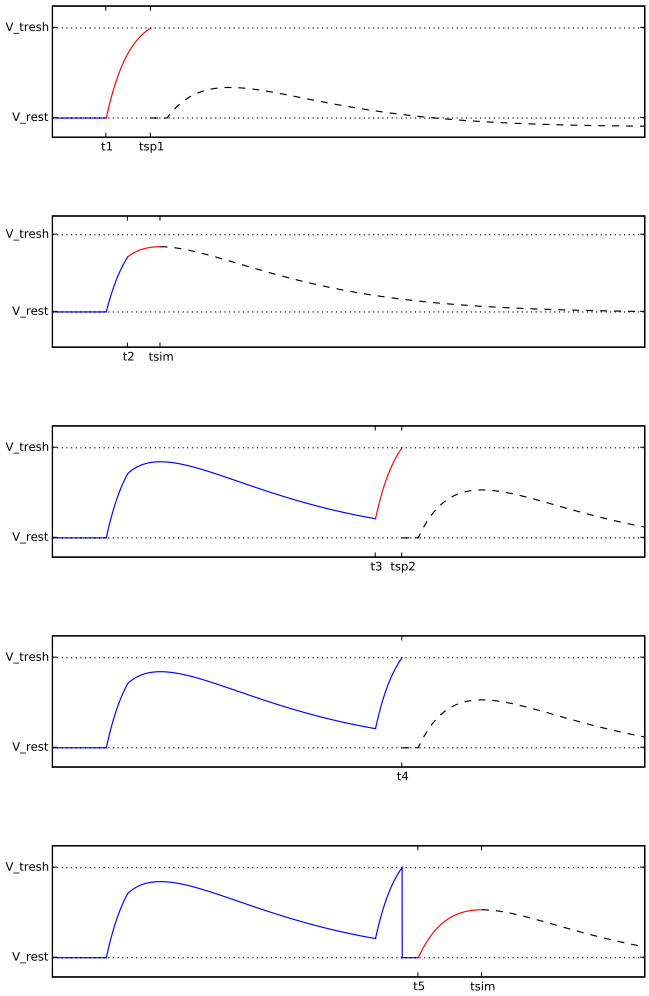


Figure 4.3: Example of Neuron Algorithm
A blue line denotes the membrane potential in the past. The red line is the course which has been simulated by the neuron algorithm, in order to calculate the timing of the next spike. Dashed line: expected course of the membrane potential.

1. At t_1 the neuron receives a spike from an excitatory synapse. A future spike is determined to happen at time $t_{\text{sp},1}$. The simulator schedules this time, to call this neuron to spike.

2. The neuron gets an *inhibitory* input at time t_2. There will be no future spike so the scheduled spike at $t_{\text{sp},1}$ is canceled.

3. *excitatory* input at time t_3. A spike for this neuron is scheduled at time $t_{\text{sp},2}$.

4. At time $t_4 = t_{\text{sp},2}$ this neuron fires and the membrane voltage is reset to V_{reset}.

5. After the refractory time has passed, the time of the next spike is calculated, in this case there is no future spike.

Before attempting the prediction of spike times, one has to determine whether the neuron will spike in the first place. The spiking case is easy to detect, one can just numerically integrate the model's differential equations until the membrane voltage V exceeds the spike detection voltage V_{Spike}. For the non-spiking case we need a condition that reliably tells us that there won't be a future spike. A careful inspection of the membrane dynamics yielded an easy and suitable way to implement this condition:
The neuron may only spike if the membrane voltage keeps increasing, i.e. $\frac{dV}{dt} > 0$.

As described above there are three basic functions needed for an event-driven neuron model to interact with the simulator. The implementation of these functions for this specific neuron model is described in the following, an example of the algorithms working is shown in figure 4.3:

After an *incoming* spike at time t_1 one first has to *update* the neuron by running algorithm 2. This algorithm just integrates the differential equations up to the current time t_1. Then the synaptic conductance is updated (in our model the synaptic conductance is increased by a value determined by the strength of the synapse), and we run algorithm 1 to calculate the *time to the next spike*. The time of the next spike $t_{\text{next,spike}}$ of this neuron is updated, in the case of no future spike it is $t_{\text{next,spike}} = \infty$.

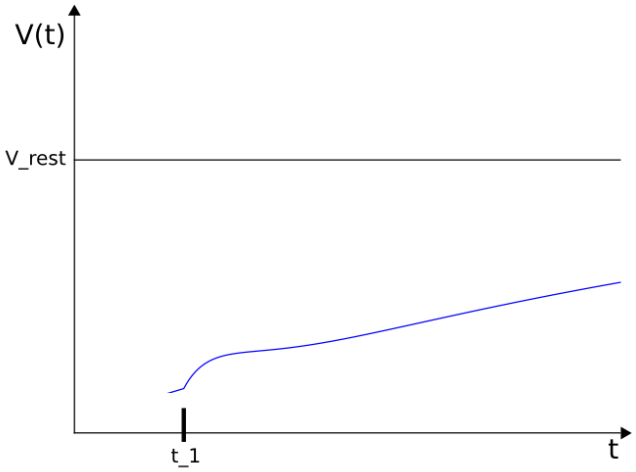


Figure 4.4: Example to illustrate the need of $\frac{dV}{dt}$ to be larger than 0, if used as a break condition to predict a future spike. An excitatory spike is received at t_1 in a hyperpolarized state (i.e. the membrane voltage V is below the resting potential V_{rest}). With no further spike the membrane voltage keeps increasing and the program would end up in an infinite loop, if it was not chosen that min $\frac{dV}{dt} > 0$.

Algorithm 1 Calculate time of next spike

while $V < V_{Spike}$ and $\frac{dV}{dt} > \min \frac{dV}{dt}$ **do**
  integrate differential equations
  $t\_sim+ = dt$
end while
if $V \geq V_{Spike}$ **then**
  return t_sim
else
  return -1
end if

Algorithm 2 Calculate equations up to current time

if $time\_to\_run < t\_sim$ **then**
  Set variables back to time of last update
end if
while $t\_sim < time\_to\_run$ **do**
  integrate differential equations
  $t\_sim+ = dt$
end while
$V_{last} = V$

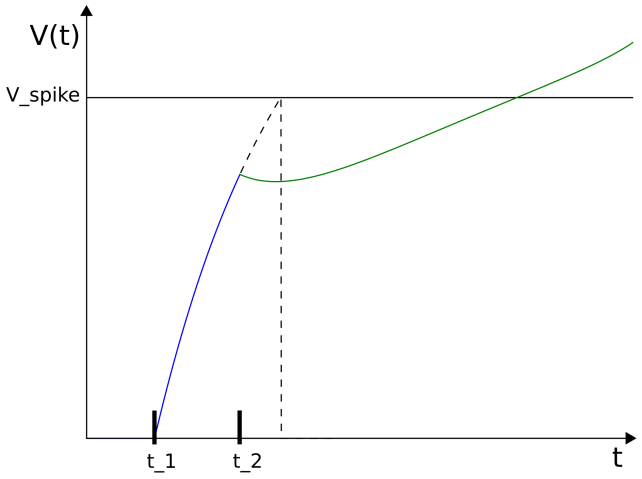


Figure 4.5: Case in which our algorithm fails to predict a spike: At t_1 the neuron receives an excitatory input, the membrane would reach the threshold if there was no other input (dashed line). At t_2 an inhibitory spike is received, the algorithm re-calculates the time of the next spike and predicts that there will not be a future spike because the derivative of the membrane voltage $\frac{dV}{dt}$ is negative at t_2. The excitatory synaptic time constant used here is 5 times the inhibitory one.

The neuron model stores two sets of variables internally, one that belongs to the time of the last update t_{last} after an incoming or outgoing spike, and one that belongs to the latest simulated time t_{sim} when calculating the time of the next spike. This way, when the simulator calls this neuron to fire, variables of t_{last} just have to be set to the ones from t_{sim}. Also, at an incoming spike, algorithm 2 checks if the current simulation time is more advanced than the latest simulated time t_{sim} and does the integration only for the missing time steps. This approach can save a lot of computation time as the numerical integration is computed multiple times for a certain time period only if an incoming spike during that period forces the model to re-calculate the next firing time.

Unfortunately, the spike prediction algorithm described above may cause erroneous results if certain values of the synaptic time constants are used. When the excitatory synaptic time constant τ_{exc} is higher than the inhibitory time constant τ_{inh} and an inhibitory spike is received after an excitatory spike the algorithm may fail to predict a spike. In figure 4.5 a case is shown in which the suggested algorithm fails to predict an outgoing spike. This problem would become more acute if multiple time constants were used in our simulated neuron model.

However, this weakness is not crucial for our neuron model, as at the moment only two different time constants are used and in most biological models inhibitory synaptic time constants are larger than excitatory ones (see e.g. *Varela et al.* [1999] and *Schneggenburger et al.* [1992]).
As in the real HICANN chip neuron circuits (denmems) can be switched together to emulate one single neuron with a high input count, for each denmem a different synaptic input behavior can be chosen. In biological terms this would be a multitude of synaptic conductances and reversal potentials. If one wants to implement this functionality in the virtual hardware, one has to optimize the spike time prediction mechanism.

Note that in algorithm 1 in the integrating while loop the break condition, which implements

the non-spiking condition, is $\frac{dV}{dt} > \min \frac{dV}{dt}$ and not $\frac{dV}{dt} > 0$, as we could end up in an infinite loop when the membrane voltage converges to the resting potential E_L, compare figure 4.4. The numerical integration of equations (2.3) is conducted with Heun's method (which is a two-stage second-order Runge-Kutta method). The step size dt and all parameters of the ADEX model can be set from the PyNN interface to the FACETS Stage 2 hardware system.

4.3 Current State of the Executable System Specification

4.3.1 Integration Into a Single Workflow

Alongside the virtual hardware, changes have been applied to the framework in which the system simulation is implemented:

- The system simulation has been taken out of the NCSim environment. As the current version of the system simulation uses SystemC and C++ only, it can run as a stand-alone program. A new top level of the virtual hardware has been created called `Stage2VirtualHardware` that instantiates the desired number of printable circuit boards and wafers.

- As now the virtual hardware can be compiled with a C++ compiler and the SystemC library, a single *makefile* has been developed that cares about all modules needed to run the executable system specification from a PyNN script, i.e. all modules belonging to the software stack for the operation of the FACETS Stage2 Hardware (cf. section 2.2) and the virtual Stage 2 Hardware can be compiled with one single makefile.

- Thus, if the Stage 2 Configurator (cf. section 2.2.4) is configured to run with the virtual hardware instead of the not yet available real hardware, the Configurator builds the desired virtual hardware and also the module that controls it (`Stage2ControlSystemSim`). The Stage2 Control also carries out all the communication with the virtual hardware: while the FPGAs and DNCs are configured via configuration packages, the configuration of all the HICANNs is bypassed, each HICANN gets access to its representation in the HardwareGraph and reads out all necessary data for its self configuration (see section 4.1). Spike events for the stimulation of neurons on a wafer are written to files which are read out during the simulation by the FPGAs. The converse procedure happens for recorded spikes, which are written to files by every FPGA.

As a consequence of all applied changes, the executable system specification is now operable from PyNN. That means that neural experiments can be set up and later run on the virtual FACETS hardware.

4.3.2 Real vs. Virtual Hardware

The current state of the system simulation represents most of the functionality of the real FACETS wafer-scale system. The similarities and differences are listed in the following:

- The virtual FACETS hardware is not operated from a separate host PC but directly on the controlling PC via configuration packets, files and function calls.

- All parts involved in the Layer 2 communication, i.e. FPGAs, DNCs and the DNC interfaces, show the same functionality and constraints (e.g. buffer sizes and clock cycle times) as in the real hardware.
- The configuration of the HICANNs is not conducted by processing packets received from a DNC but via direct access to the GraphModel. However, the HICANN considers all information actually available in the HardwareGraph.
- The routing of spikes on the wafer does not involve any timing. This means that a spike released by one HICANN instantaneously reaches its target neurons.
- Most of the functional units of the ANNCORE (e.g. synapse drivers, neuron circuits) have a sufficient representation in the virtual hardware. Priority Encoders, the STDP mechanism and on-chip spike sources (which are not yet supported by the MappingTool) still lack a sufficient level of specificity.

Under these conditions the virtual hardware exhibits most of the functionality and constraint of the real hardware. The influence of this has been further tested in chapter 6.

5 Transformation of Biological Parameters into their Hardware Equivalents

Unfortunately, finding an adequate biology-to-hardware translation is not trivial, as not every biological parameter has its individual counterpart on the micro chip, and often is emulated by a set of correlating parameters. Many of these values can not be configured individually for every unit in the chip but only for a group of units. Due to process limitations, hardware parameters are limited to a certain range of possible values. However, the chip is designed to provide a hardware representation of all biologically relevant parameter values.

The transformation of parameters takes place in the MappingTool(see section 2.2.3), after the placement of neurons and the routing of synaptic connections was conducted. For every HICANN, the class *CustomParamterTransformation* is instantiated with access to all nodes and connections of both the biological and the hardware GraphModel that are needed to perform the transformation for one HICANN.

5.1 Synapse Drivers and Synaptic Weights

As described in section 2.1.2.2, not every synapse can be configured individually. For every synapse row a fixed maximum conductance $g_{\max}$ can be set at the synapse driver. Every synapse stores a 4-bit weight with values in the range of $[0, 15]$, resulting in an applied weight in the range of $[\frac{0}{16}, \frac{15}{16}] \cdot g_{\max}$. Hence, there is a maximum of 16 different settings of synaptic weights per row. Remember that the output signal of a synapse is a square current pulse with length τ_{stdf} and amplitude $weight \cdot g_{\max}$. This current ultimately modulates the synaptic conductance between the neuron's membrane capacitance and the dedicated reversal synaptic potential. Therefore, the biologic synaptic conductance needs to be mapped to the hardware circuitry by the following transformation:

$$g_{\text{bio}} \rightarrow g_{\max} \cdot weight \cdot \tau_{\text{stdf}} \tag{5.1}$$

Short-term plasticity affects the maximum conductance of a synapse. On the hardware, this is realized by varying the pulse length τ_{stdf}. As τ_{stdf} has no effect on the initial configuration, we consider it as a scaling factor and set it to 1 for further calculations. This just leaves $g_{\max}$ as an adjustable parameter. The current parameter transformation offers three different ways of transformation:

- One that scales $g_{\max}$ such that the largest biological value of this row is exactly mapped to the highest settable digital weight (15).
  $g_{\max} = \frac{16}{15} \cdot g_{\text{bio,max}}$
- One that calculates the mean biological weight per row and maps it to the half maximum digital weight.
  $g_{\max} = 2 \cdot g_{\text{bio,mean}}$

This has the consequence that values higher than $2g_{\text{bio,mean}}$ are capped to a maximum weight of 15.

- One that just exploits a part of the range of settable digits. Like in the first case, we take the largest biological value and now map it to an arbitrary digital weight, e.g. the half maximum:
  $g_{max} = 2 \cdot g_{\text{bio,max}}$.
  The use of this transformation may be useful, when STDP is integrated so that both long-term potentiation and depression is possible for all connections.

In the next step, the digital 4-bit weights are calculated as the fraction of the biological value g_{bio} and $g_{\max}$ multiplied with 16.

$$weight(i) = stochastic\_round\left(\frac{g_{\text{bio}}(i)}{g_{\max}} \cdot 16\right) \tag{5.2}$$

The obtained values are rounded to values between 0 and 15. In order to avoid a systematic attenuation or strengthening of synaptic connections, *stochastic rounding* is used: e.g. a value of 7.4 is rounded off with a probability of 0.6 and rounded up with a probability of 0.4. Thereby it is assured, that the total and local (e.g. for one synaptic row) mean of synaptic weights remains constant.

When it comes to choose one of the types of configuration presented above, there is no universal solution for all given network topologies. One has to know the optimization target, which can be for example a very precise parameter transformation for certain values or a transformation, that is not that precise, but gives an ample scope for a long-term evolution of synaptic weights, when STDP is used.

As a possible alternative the mapping and routing algorithms in the *MappingTool* could also take into account the weight of synaptic connections, and thereby try to place the neurons and route the connections such that synapses with similar weights are emulated in circuits of one row in the synapse block. The hardware implementation also allows to combine two synapse circuits, which are located in the same column and are operated by the same synapse driver, to represent one synapse with an 8-bit resolution. This functionality is not yet supported by the mapping and routing algorithms and also would result in a halving of total possible inputs per neuron.

5.2 Short-Term Plasticity

Among others, PyNN supports the short-term plasticity mechanism presented in *Markram et al.* [1998] (name in PyNN: `TsodyksMarkramMechanism`). The FACETS Hardware implements a slightly modified version of it: it can not implement depression and facilitation at the same time. The basic functionality, however, stays the same:
The strength of the synaptic response at an incoming spike is limited by its absolute synaptic efficacy $w_{textmax}$, the currently available *recovered partition* R, and the *utilization of synaptic efficacy* U. The normalized absolute synaptic efficacy can be divided into two parts: the *recovered partition* R and the *inactive partition* I:
$1 = R + I$
Only the recovered partition R is available for and only the fraction U of the recovered partition affects the strength of a synaptic response, which therefore is equal to $w_{\max} \cdot R \cdot U$.

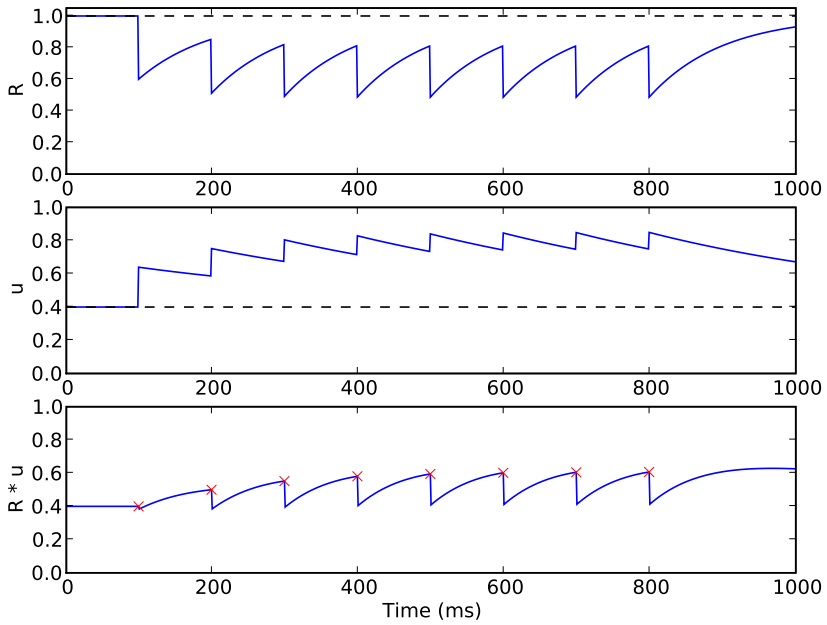


Figure 5.1: Example of the short-term plasticity mechanism by *Markram et al.* [1998] as described in text. This figure shows the impact of a pre-synaptic spike train onto the variables determining the plasticity mechanism. In the top the *recovered partition* R decreases with every AP, and then recovers with τ_{rec}. In the middle the *usable synaptic efficacy* u is shown, it increases with every AP and then decays with τ_{facil} back to its resting value U. At the bottom, the product of R and U is plotted, the value of which determines the amplitude of the synaptic response at arrival times of incoming spikes. One can see that the applied amplitude of the synaptic response (denoted with a red cross) increases with every AP and converges to a maximum value, thus this synapse shows *facilitating* behavior. (Parameters used for this simulation: $U = 0.4$, $\tau_{\text{rec}} = 100$ms, τ_{facil}=400 ms)

After a synaptic response to an incoming spike the fraction U of the recovered partition gets instantaneously unavailable, and recovers with time constant τ_{rec}. In other words, the amount of $U \cdot R$ is transferred from the recovered to the inactive partition. R then recovers with τ_{rec} from I.

A facilitation mechanism is introduced by a variation of the usable synaptic efficacy U, the running value of the utilization of synaptic efficacy is referred to as $u(t)$. With every AP $u(t)$ is increased by a certain value. The running usable efficacy $u(t)$ then decays with the time constant τ_{facil} to its resting value U. The amplitude to be added to the current utilization u was specified as $U \cdot (1 - u)$, ensuring that $u < 1$.

In Figure 5.1 this model of short-term plasticity was stimulated with a pre-synaptic spike train. The course of the variables u and R, as well as the strength of the resulting synaptic response is shown.

At an incoming spike at time t_{AP} the following happens:

$$R \to R \cdot (1 - u) = R(t_{\text{AP}})$$
$$u \to u + U \cdot (1 - u) = u(t_{\text{AP}})$$

The course of $R(t)$ and $u(t)$ at time Δt after t_{AP} are described by the following equations:

$$\begin{aligned} R(t_{\mathrm{AP}} + \Delta t) &= 1 - (1 - R(t_{\mathrm{AP}})) \cdot \exp \frac{-\Delta t}{\tau_{\mathrm{rec}}} \\ u(t_{\mathrm{AP}} + \Delta t) &= U + (u - U(t_{\mathrm{AP}})) \cdot \exp \frac{-\Delta t}{\tau_{\mathrm{facil}}} \end{aligned} \tag{5.3}$$

The amplitude of a synaptic response generated after an incoming action potential is given by:

$$w = w_{\max} \cdot R \cdot u \tag{5.4}$$

Whether a synapse is facilitating or depressing depends on the time constants, the usable efficacy U and the stimulation frequency, also a mixed behavior is possible, e.g. facilitation at the beginning of high pre-synaptic activity and depression later on.
As the circuits in the synapse drivers of the FACETS hardware system support either depression or facilitation, but not a combination of both (cf. 2.1.2.3), a restriction is given for the translation of the parameters given in the `TsodyksMarkramMechanism`, as one should use only one of the two time constants τ_{rec} and τ_{facil} and set the other one to *zero*.

5.2.1 Depression Mode

For pure depression mode ($\tau_{\mathrm{facil}} = 0$) the applied synaptic weight in the original model from Tsodyks and Markram is:

$$w = w_{\texttt{max}} \cdot R \cdot U \tag{5.5}$$

The short-term plasticity circuits in the hardware modulate the synaptic response in the following way, as described in equation (2.1):

$$w = w_{\mathrm{max,HW}} \cdot (1 - \lambda \cdot I) \tag{5.6}$$

Remember that the scaling factor λ is determined by the product of the gain factor μ of the used operational transconductance amplifier and the voltage V_{max}. After every pre-synaptic AP, the inactive partition I is increased by $(1 - I) \cdot \frac{C_2}{C_2 + C_1}$ and decays with a time constant regulated by the current I_{rec}. One can see that in the original model only the usable efficacy U is applied, much in contrast to the hardware implementation, where this is not taken into account. Therefore, one has to scale the maximum weight $w_{\mathrm{max,HW}}$ accordingly (or rather: the maximum conductance per synapse row $g_{\max}$, as described above). The course of R can be exactly represented by $(1-I)$ if the *scaling factor* λ of the short-term plasticity mechanism is set to 1. Hence, one has to perform appropriate translations from parameters of the original model defined in PyNN into corresponding hardware counterparts.

$$\begin{aligned} R &\rightarrow (1 - \lambda \cdot I) \\ w_{\max} \cdot U &\rightarrow w_{\mathrm{max,HW}} \\ \tau_{\mathrm{rec}} &\rightarrow I_{\mathrm{rec}} \end{aligned} \tag{5.7}$$

The recovery time constant τ_{rec} can be directly transferred by adjusting the current I_{rec} accordingly, for the quantized changes of I (in the model of Markram: utilization of synaptic efficacy U) the best matching hardware configuration of C_1 and C_2 is chosen (possible values:

$[\frac{1}{9}, \frac{3}{11}, \frac{5}{13}, \frac{7}{15}]$). The resulting values for the corresponding voltages, currents and bits are listed in Table 5.1.

Parameter Transformation in Depression Mode

hardware parameter	**transformation**
$V_{\max}$	$\frac{1}{\mu}$
I_{rec}	$\text{const} \cdot \tau_{\text{rec}}$
$\frac{C_2}{C_2+C_1} \in [\frac{1}{9}, \frac{3}{11}, \frac{5}{13}, \frac{7}{15}]$	U
$w_{\max}$	$w_{\max,\text{PyNN}} \cdot U$
λ	1

Table 5.1: Parameter Transformation from the original model to a hardware configuration for *Short-Term Depression*

5.2.2 Facilitation Mode

The transformation task gets more complicated when it comes to *facilitation*. If τ_{rec} is set to *zero* in the original model, the recovered partition R is fully available all the time (i.e. $R = 1 = \text{const}$) and only the utilization of synaptic efficacy u varies in time. Thus, the resulting applied weight is:

$$w = w_{\max} \cdot u \tag{5.8}$$

The synaptic response implemented in the hardware system according to (2.2), with $N = \frac{V_{\text{fac}}}{V_{\max}}$, is:

$$w_{\text{HW}} = w_{\max,\text{HW}} \cdot (1 + \lambda \cdot (I - N)) \tag{5.9}$$

Therefore, the course of u has to be emulated by $(1 + \lambda \cdot (I - N)$. There are two important constraints: we have to assure that the resulting weight w_{HW} can not exceed w_{max} given in PyNN, and $\lambda \cdot (I - N)$ must not exceed 1 as the operational transconductance amplifier OTA2 (cf. figure 2.3) does not support an output current higher than its bias (cf. 2.1.2.5). The second condition can be fulfilled by setting $V_{\max} = \frac{1}{\mu}$. Note that at the first AP the inactive partition is $I = 0$, so that the applied weight will be:

$$w_{\text{HW}} = w_{\max,\text{HW}} \cdot (1 - N)$$

With this setting of $V_{\max}$, a possible range of $[1 - N, 2 - N] \cdot w_{\max,\text{HW}}$ for applied weights is possible. The original model is capable of a range in between U and 1, so that the following equations have to be solved in order to project the full range of possible biological values for U onto the hardware.

$$\begin{aligned} (1 - N) \cdot w_{\max,\text{hw}} &= U \cdot w_{\max} \\ (2 - N) \cdot w_{\max,\text{hw}} &= 1 \cdot w_{\max} \end{aligned} \tag{5.10}$$

U and $w_{\max}$ are given, so that we get the following solution for $w_{\max,\text{HW}}$, N and thus V_{fac}.

$$\begin{aligned} w_{\max,\text{HW}} &= (1 - U) \cdot w_{\max} \\ N &= \frac{1 - 2 \cdot U}{1 - U} \\ \Rightarrow V_{\text{fac}} &= \frac{1 - 2 \cdot U}{1 - U} \cdot V_{\max} \end{aligned} \tag{5.11}$$

This transformation allows us to imitate the behavior of short-term potentiation with the dynamics implemented in the HICANN chip, the resulting electrical values are listed in Table 5.2.

Parameter Transformation in Facilitation Mode

hardware parameter	**transformation**
V_{max}	$\frac{1}{\mu}$
V_{fac}	$N \cdot V_{\text{max}}$
I_{rec}	$\text{const} \cdot \tau_{\text{facil}}$
$\frac{C_2}{C_2+C_1} \in [\frac{1}{9}, \frac{3}{11}, \frac{5}{13}, \frac{7}{15}]$	U
$w_{\text{max,HW}}$	$(1-U) \cdot w_{\text{max}}$
λ	1
N	$\frac{1-2U}{1-U}$

Table 5.2: Parameter Transformation from the original model to a hardware configuration for *Short-Term Facilitation*.

5.2.3 Considering Restrictions from Chip Design

The application of the short-term plasticity mechanism, including all necessary parameter transformations (presented in the paragraphs above), works well (as is also seen in section 6.3.2), as long as the mechanism is configurable for every single synapse. In reality this is not the case, many constraints have to be taken into account:

- The electrical entities V_{max}, V_{fac} and I_{rec}, which are essential for the STP dynamics, are shared by a block of 64 synapse drivers.
- In every synapse driver one can set just one mode of short-term plasticity, adjustable parameters are the `enable` and `mode` bits as well as two bits to set the size of the capacitor C_2, responsible for the size of the *utilization of synaptic efficacy*.
- Voltages are restricted to values between 0V and 1.8V, currents to values between 100nA and 20 μA, both can be set with 10-bit precision. The gain factor μ is not manually adjustable.
- All synapses (up to 512) that are operated by one synapse driver share one setting of short-term plasticity. For every pre-synaptic source, only one capacitor emulating the inactive partition I is provided.

STP-compatible mapping and routing algorithms have to consider all these restrictions and draw synaptic connections in a way that synapses with a similar short-term plasticity behavior are fed by the same synapse driver. At the current point of development, this feature is not yet supported by the *MappingTool*.

In order to realistically represent the capabilities of the FACETS Stage 2 system, the following approximations and simplifications are made in the process of parameter transformation.

1. Voltages and currents can be set individually for every synapse driver (of course, this is only possible in the executable system simulation). Thus, a future setup is resembled, when synapses will be grouped appropriately by the *MappingTool*.

2. For all synapses belonging to one synapse driver, the biological parameters (i.e. U, τ_{rec} and τ_{facil}) defined in the *PyNN* script are collected and the mean values are computed.

3. From the mean values it is evaluated whether short-term plasticity shall be enabled and which mode shall be used.

4. The transformation is performed according to the translations methods presented above.

In fact, this sequence may result in critical distortions of the network dynamics, which have to be evaluated and compared to results obtained with established software simulators.

5.3 Membrane Circuits

The PyNN description of the adaptive exponential integrate-and-fire neuron model (ADEX, see 2.1.2.7), which is implemented in the HICANN chip, holds a total of 18 parameters. The *denmem* circuits in the HICANN foresee 24 for parameters for the hardware implementation of this model. This means, that not every model parameter in PyNN has its individual counterpart in the hardware but can be represented by several hardware parameters. Also, not for every parameter there is a linear transformation, instead, a look-up table has to be used when performing the parameter transformation. These look-up tables first have to be determined from simulations of the membrane circuits. The HICANN chip is still under test and a suitable calibration is not yet available.

At the moment, the *virtual* FACETS hardware is the only instance that is operated by the PyNN.hardware.stage2 module (cf. section 2.2.1). Thus, for now, it is sufficient to provide the transformation for the virtual hardware, which exactly implements the ADEX model. Therefore, the parameters of a biological neuron are just passed through to its counterpart in the executable system simulation. Furthermore, every simulated hardware neuron (see 4.2.3) is configured with the *speedup factor* it shall run with compared to biological time, with the integration *time step dt* it shall use for numerical integration and with a flag, if its membrane is to be *recorded* or not.

6 Experiments and Results

In this chapter, *experiments* conducted with the FACETS Stage 2 executable system specification are presented. The chapter is mainly divided into two parts: *low-level* tests of submodules of the system simulation (e.g. neuron models or short-term plasticity) and the application of the FACETS Demonstrator benchmark models presented in chapter 3. Before, the experimental setup of the executable system specification and the operating workflow are described.

6.1 Experimental Workflow

Every *experiment* starts with a PyNN description of a certain neural experiment, which contains the network model, stimulation data and all the information related to the experiment. The PyNN interface to the FACETS Stage 2 Hardware *PyNN.hardware.stage2* converts the network model into a graph representation (the so-called BioGraph, cf. 2.2.2), which is then passed to the MappingTool (cf. section 2.2.3). The MappingTool first places the biological neurons onto the available resources of FACETS Hardware. The utilized algorithm can be chosen from e.g. *random placement*, the *NForceCluster* algorithm or an *external* manual placement. Next, the *routing* of synaptic connections on and off the wafer is performed, followed by the *transformation* of biological *parameters* into their counterparts on the hardware. The configuration of the hardware system is extracted from the Hardware Graph and communicated to the corresponding modules of the executable system specification. Input spike trains, created from the PyNN description of the experiment, are preprocessed for the neuromorphic hardware, for example, they have to be transformed to the time scale of the accelerated hardware. This spike data is written to files, which are later read from the virtual FPGAs.
The executable system specification is then called to run the simulation for the desired time. The FPGAs start to read spike stimuli which are then transmitted the HICANNs, where neurons are stimulated thus initiating the network dynamics. Pulse events of neurons which were set to be traced, are recorded and transferred up to the FPGAs, where they are written into another file.
After the simulation has finished, these spikes are translated back into their biological time scale and returned to the PyNN program that initiated the whole procedure. The retrieved data can now be analyzed by the user.

6.2 General Experimental Setup

Before a simulation can be started, it has to be decided how much of the hardware shall be utilized and thereby simulated. Of course, more hardware units, e.g. more FPGAs, DNCs or HICANNs, imply more resources and more features that are used and tested in an experiment. Certainly, this results in an increase of simulation time. As a compromise it was decided to

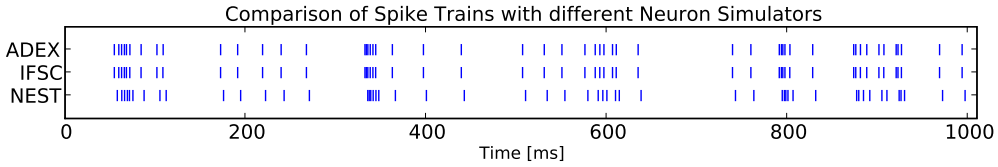


Figure 6.1: Spike trains of a single leaky integrate-and-fire neuron simulated with NEST and with the virtual hardware. The simulation on the virtual hardware was conducted once with the exact integrate-and-fire neuron (IFSC, see 4.2.3.1) and once with newly developed event-based ADEX neuron (see 4.2.3.2). The neuron was stimulated with a Poisson Input of 50Hz.

use the following setup as default:

- 1 FPGA
- 4 DNCs
- 4 reticles with 8 HICANNS each
- a maximum of 64 neurons per HICANN

This results in a maximum of 2048 neurons available on the system. The speedup factor with which neurons and synapses are emulated in analog and mixed-signal circuits compared to biological real time was chosen to be 10^4, which is also supposed by the real wafer-scale system. The default time step used for the approximation method in the ADEX neuron model (see section 4.2.3.2 was $dt = 0.01$ms (biological time). If not explicitly mentioned otherwise, this setup was used for all experiments described in this chapter.

6.3 Low-Level Experiments

Before running large neural network models with the executable system simulation, one first wants to test the behavior of all participating submodules in order to find their intrinsic flaws and constraints. Especially the enhanced modules of the wafer-scale system simulation (cf. chapter 4) are tested. Furthermore, also the maximum reachable bandwidth of pulse events over the Layer 2 communication was estimated. For every comparison made between a hardware emulation (with the executable system specification) and a software simulation (with either NEST or NEURON), only one line of code was changed in the PyNN description of the experiment, namely the one which specifies the simulator to use.

6.3.1 Neuron Test

As described in section 4.2.3, two neuron models were implemented in the behavioral model of the ANNCORE. The first is a leaky integrate-and-fire model with synaptic conductances

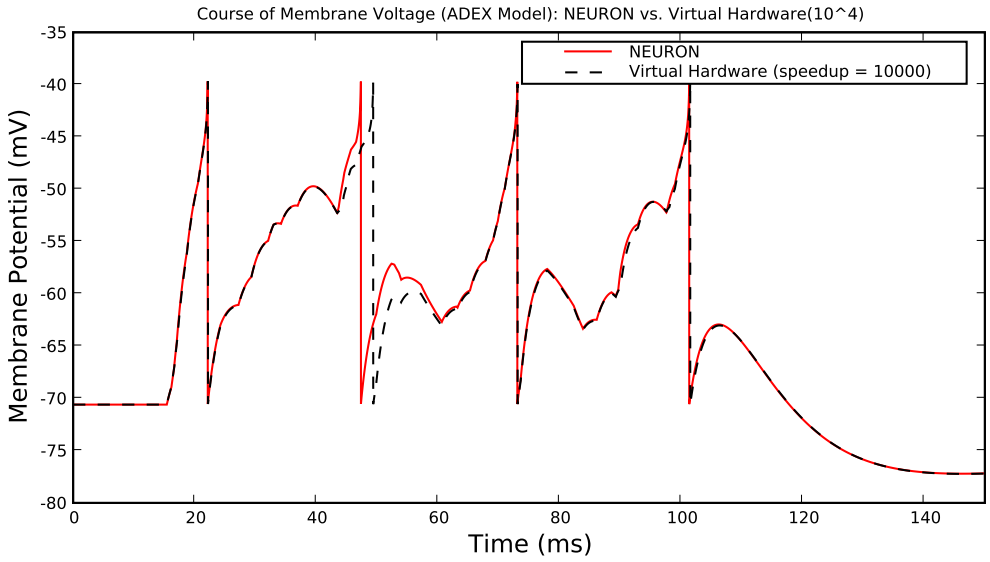


Figure 6.2: Membrane potential of an adaptive exponential integrate-and-fire neuron model stimulated with excitatory and inhibitory Poisson input. The voltage course of both the NEURON simulation and the simulation on the virtual hardware is shown. The used speedup factor was 10^4.

(IFSC, see 4.2.3.1), which calculates the spikes and neuron variables exactly up to machine precision. The second is an event-based version of the ADEX neuron model (see 4.2.3.2), which uses an approximation method. The ADEX model can also be used as a leaky integrate-and-fire model if the adaption current and the exponential term are switched off, thus, the spike behavior of both IFSC and ADEX can be compared with a leaky integrate-and-fire model simulated with a software simulator. Such a comparison is illustrated in figure 6.1. One can recognize almost the exact same spike pattern. The spike trains generated with the executable system specification start a bit earlier than the ones produced by NEST. This happens because the translation between the hardware- and the biological domain of the spike output from hardware to biological time is not yet fully matured.

Subsequently, the correct behavior of the event-based ADEX neuron model was tested: A single ADEX neuron (`EIF_cond_exp_isfa_ista`) with synaptic conductance was stimulated with 500Hz excitatory and 200Hz inhibitory Poisson input. In figure 6.2 the time course of the membrane voltage of this neuron is shown, simulated with both NEURON and the virtual hardware. The trace produced by the virtual hardware resembles the one recorded with NEURON with the only exception that the second spike occurs a bit later. The reason for this can be found by looking at the course of the excitatory conductance of the neuron, shown in figure 6.3. One can recognize that some spikes (between 43ms and 47ms) in the system simulation arrive later than in the corresponding NEURON simulation. This happens particularly when lots of spikes lay within a very short time interval. The reason for this is the method with which spikes are transmitted from the FPGA to the HICANNs over Layer 2 connections. Spikes cannot not be delivered at an arbitrary time but have to be processed one after another. In the case of a *burst*, some spikes can not be delivered on time and arrive late at their targets (i.e. at the Layer 1 network and thus at the neurons). It can even happen that spikes are lost, when too many pulse packets have to be transmitted at the same time.

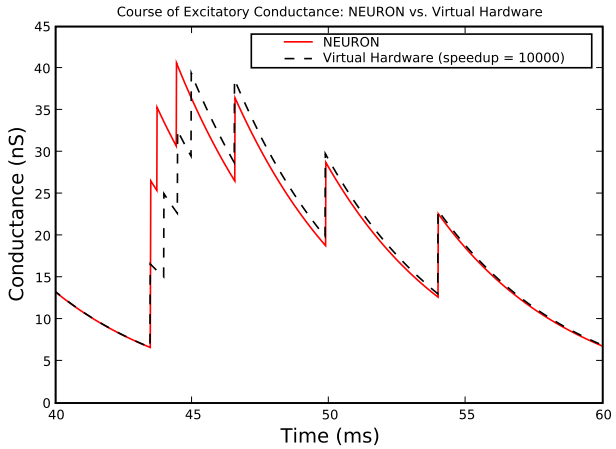


Figure 6.3: Excerpt of the course of the excitatory conductance of the ADEX as simulated in figure 6.2. One can see that some spikes arrive later in the virtual hardware simulation than in the NEURON simulation.The explanation for that is given in the text.

This limitation is further examined in section 6.3.3.

This problem can be alleviated if a lower speedup factor for the membrane circuits is used in the system simulation. Remember that the simulated neuron used in the example above works with an acceleration factor of 10^4 compared to biological time, i.e. 1s of biological time corresponds to 0.1ms in hardware time. In a subsequent run, the speedup factor used for the simulation was reduced to 10^3. Note that the digital units (i.e. the modules involved in the Layer 2 communication) of the system still run with the same speed. Consequently, the total simulation time is increased by an order of magnitude. The result can be seen in figure 6.4, where the membrane trace of the ADEX neuron coincides with the one from NEURON. Thus, it has been shown that the simulated hardware neurons can perform like their equivalents in software simulators.

6.3.2 Short-Term Synaptic Plasticity

In the following the *short-term plasticity mechanism* implemented in the virtual hardware (see section 4.2.2) and thus the parameter translation developed in section 5.2 is tested. The setup is the following: a spike source (`SpikeSourceArray`) is connected to a simple integrate-and-fire neuron (`IF_cond_exp`) with a short-term plasticity mechanism (`TsodyksMarkramMechanism`). The strength (i.e. the weight) of the connection is chosen to be 50nS. This means that the synaptic conductance of the neuron is increased by this value at every post-synaptic spike, if the short-term plasticity mechanism is disabled. If STP is enabled, the synaptic conductance is increased by just a fraction of this weight (the maximum weight could then be viewed as an absolute usable efficacy). Both the (excitatory) conductance and the membrane potential are recorded.

6.3.2.1 Depression

The parameters used in PyNN for the depression mode are the following: $U = 0.4$, $\tau_{\text{rec}} = 400\text{ms}$ and $\tau_{\text{fac}} = 0$ (These parameters are also used in the Layer 2/3 Attractor Memory

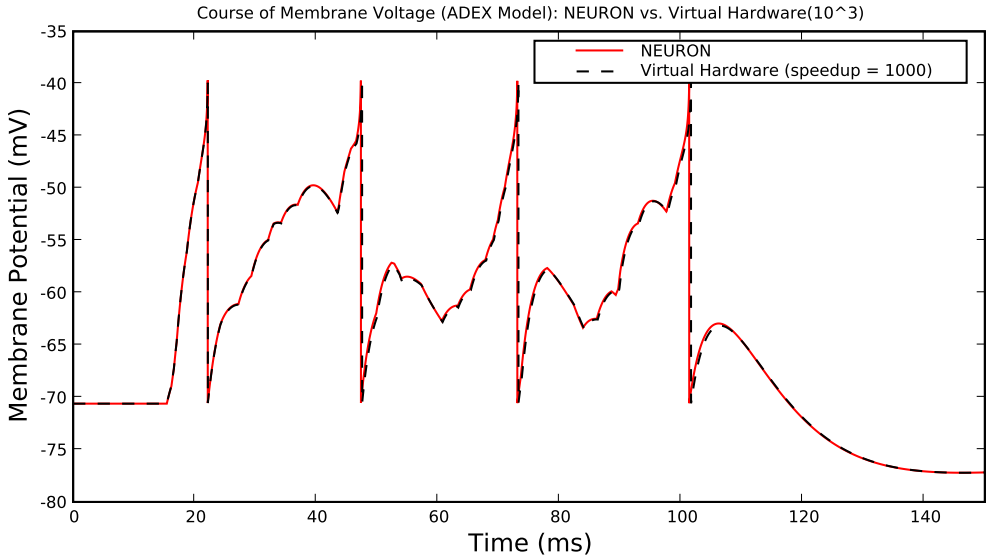


Figure 6.4: Membrane potential of the ADEX model simulated with NEURON and the virtual hardware (with a speedup factor of 10^3). Same synaptic input as in figure 6.2.

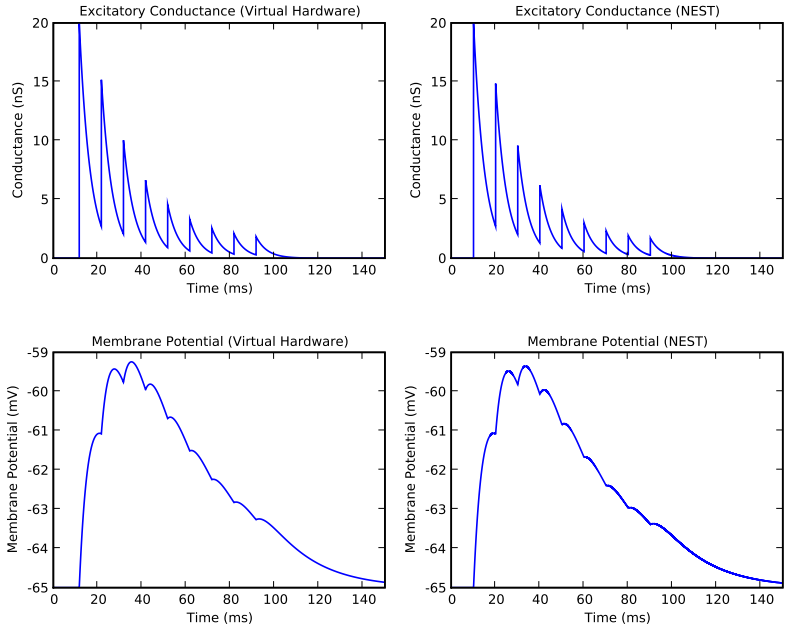


Figure 6.5: Example the *short-term depression* on the virtual hardware compared to a simulation with NEST

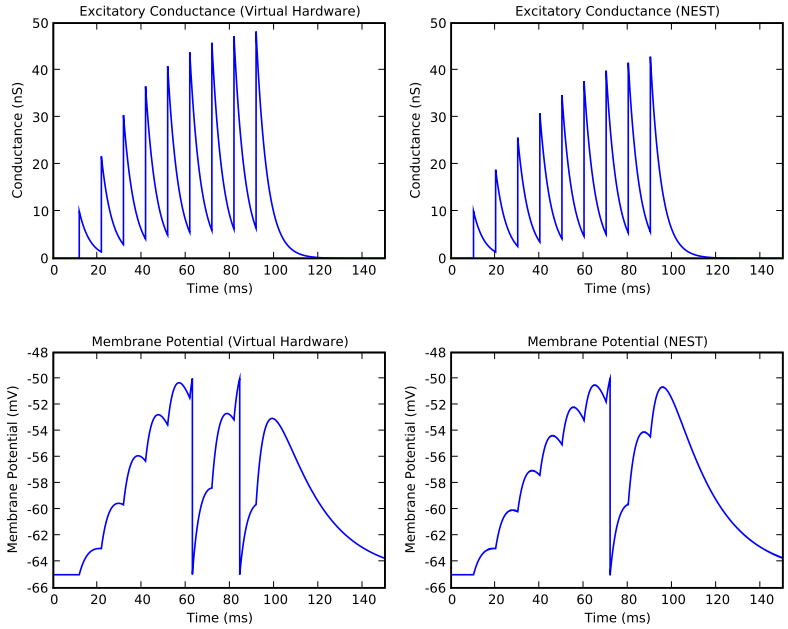


Figure 6.6: Example of the *short-term facilitation* mechanism of the virtual hardware compared to a simulation with NEST.

network model described in section 3.1.1). For an interpretation of these values, see section 5.2, where the short-term plasticity mechanism according to *Markram et al.* [1998] is described. Remember that for the usable efficacy U in the hardware only 4 values are possible ($[\frac{1}{9}, \frac{3}{11}, \frac{5}{13}, \frac{7}{15}]$). In this example, this results in the following distorted usable efficacy $U_{\mathrm{SE}} = \frac{5}{13} \approx 0.385$ in the hardware. In figure 6.5 the influence of the short-term plasticity mechanism on the neuron's membrane potential and the excitatory conductance are plotted for the virtual hardware (on the left) and for NEST (on the right). One can see that the amplitude of the increase of the conductance decreases with every subsequent incoming spike. While the first quantal increase has an amplitude of 20nS, the last one is only about 2nS high. The course of the membrane potential of the neuron simulated with the virtual hardware resembles the one simulated with NEST very well.

6.3.2.2 Facilitation

For the case of short-term depression, the following parameters were used in the PyNN script: $U = 0.2$, $\tau_{\mathrm{rec}} = 0$ and $\tau_{\mathrm{fac}} = 100\mathrm{ms}$, thus the corresponding usable efficacy in the hardware being $U_{\mathrm{SE}} = \frac{3}{11} = 0.273$.(Again, the parameter were taken from Layer 2/3 Attractor Memory network model) The resulting conductance and membrane traces of the simulation with the virtual hardware and NEST can be compared in figure 6.6. We can see that the facilitation mechanism in the executable system specification works: the quantal increment of the synaptic conductance increases with every subsequent incoming spike. However, the course of both the conductance and the membrane potential differ from the ones produced by NEST. This happens mainly because the usable efficacy in the virtual hardware is distorted: $U_{\mathrm{SE}} = 0.273$

instead of $U = 0.2$ due to the transformation algorithm described in section 5.2.2.

It was successfully shown in this section that the STP mechanism implemented in the synapse drivers of the virtual ANNCORE as well as the parameter transformation work well, with the constraint that for the usable efficacy U only 4 values are possible. Maybe the best way to avoid this distortion would be to restrict U to these 4 possible values already in the PyNN interface to the FACETS Hardware.

6.3.3 Bandwidth Test

When attempting to simulate large networks, such as the FACETS Demonstrator benchmark models, it was noticed that the neurons did not receive the expected input. Furthermore, not every spike generated in the simulated ANNCOREs was recorded from the FPGA. The reason for this was easy to find: a limited bandwidth for the transport of Layer 2 Pulse packets between FPGA, DNCs and HICANNs.

Therefore, a systematic examination of the bandwidth was conducted. For that, a single neuron was stimulated by external Poisson spike sources while varying their input rate. It is important to know how the MappingTool arranges external spike sources on the wafer: Every DNC Interface on a HICANN contains 8 Layer 2 - to - Layer 1 channels. Each channel is connected to a horizontal Layer 1 bus of the HICANN. The direction of the channel can be set to either *towards* the HICANN (for the sending of spikes) or *from* the HICANN (for the recording of spikes). Up to 64 neurons can send on such a L2-to-L1 channel.
External spike sources are distributed in the following way onto these channels on a wafer (in this setup: a grid of 8×4 HICANNs):
At first, one channel of the first HICANN is filled up with 64 source neurons. The same is subsequently repeated with all other HICANNs. When the first channel of the last HICANN is full, the algorithm proceeds with the second channel on the first HICANN and so on. A sweep over the number of Poisson spike sources and the Poisson rate was conducted, while each time counting the number of spikes received by the neuron. Thereby the following limitations for Layer 2 connections could be determined: The maximum bandwidth between a DNC and

Maximal Bandwidth of Neural Events over Layer 2

Connection	**bandwidth (in spikes/s)**
$1\ FPGA \rightarrow 1\ DNC \rightarrow 1\ HICANN$	2.0×10^7
$1\ FPGA \rightarrow 1\ DNC \rightarrow 8\ HICANN$	9.4×10^7
$1\ FPGA \rightarrow 4\ DNC \rightarrow 8\ HICANN$	3.3×10^8

Table 6.1: Maximum reachable bandwidth of spike events over Layer 2 communication

one HICANN is 2.0×10^7 spikes/s. One DNC can handle up to 9.4×10^7 spikes/s, when connected to several HICANNs. An FPGA, which is connected to 4 DNCs with 8 HICANNs each, reaches a maximum of 3.3×10^8 spikes/s (see table 6.1).

The resulting maximum total input rates for one FPGA depending on the used speedup factor are listed in table 6.2. When preparing a model to be run on the FACETS Hardware, these limitations should be considered. Also, the spike sources should be distributed onto all available HICANNs for a maximization of input bandwidth.

Maximal input rate per FPGA for different speedup factors	
speedup factor	**maximal total input rate**
10^3	3.3×10^5
10^4	3.3×10^4
10^5	3.3×10^3

Table 6.2: Maximum available input rate for one FPGA using different speedup factors in the ANNCORE

6.4 High-Level Experiments with the FACETS Demonstrator Benchmarks

In this section, first simulation results of the FACETS Demonstrator benchmarks models (as described in chapter 3) are presented. In order to run these models on the virtual hardware, some modifications had to be applied. For all simulations a speedup factor of 10^3 was used to ensure a Layer 2 bandwidth that is high enough for an acceptable stimulation and recording of all neurons. A qualitative profiling of the computational performance has been carried out for the first benchmark model in section 6.4.1.3.

6.4.1 KTH Layer 2/3 Attractor Memory

6.4.1.1 Modifications Applied to the Default Model

The following changes were made to the Layer 2/3 Attractor Memory Network model described in section 3.1.1:

- The architecture was reduced to 6 hypercolumns with 3 minicolumns each, resulting in a total number of 624 neurons.

- The original model uses the `IF_cond_exp_gsfa_grr` neuron model for the pyramidal and RSNP cells. This model is based on *Muller et al.* [2007] and adds a conductance-based adaptation as well as a relative refractoriness mechanism to the conductance-based leaky integrate-and-fire neuron. Unfortunately this model is not (yet) supported by the FACETS hardware and a transformation to the ADEX model is not close at hand, therefore it was replaced by a simple leaky IF model (`IF_cond_exp`) with an absolute refractory time ($\tau_{\text{refrac}} = 1.0\text{ms}$).

- In the default model every pyramidal cell receives a Poisson input with a frequency of 4000Hz. This results in a total stimulation rate of $2.16 \cdot 10^6$Hz for the whole network containing 540 pyramidal cells. Remembering that the maximum total bandwidth for one FPGA at a speedup factor 10^3 is $3.3 \cdot 10^5$ (see table 6.2, the input rate for one cell was reduced to $\frac{4000}{7} = 571$Hz, at the same time the synaptic weight of the Poisson sources was multiplied by 7. Furthermore, instead of one spike source with 571Hz, every neuron has a higher number of spike sources with a reduced rate, such that the

spike sources are distributed over all HICANNs (cf. section 6.3.3). In this case, every pyramidal cell receives individual Poisson input from 7 spike sources with 81.5Hz each.

- All synaptic delays were set to 0, as delays are not supported by the virtual hardware.

6.4.1.2 Results

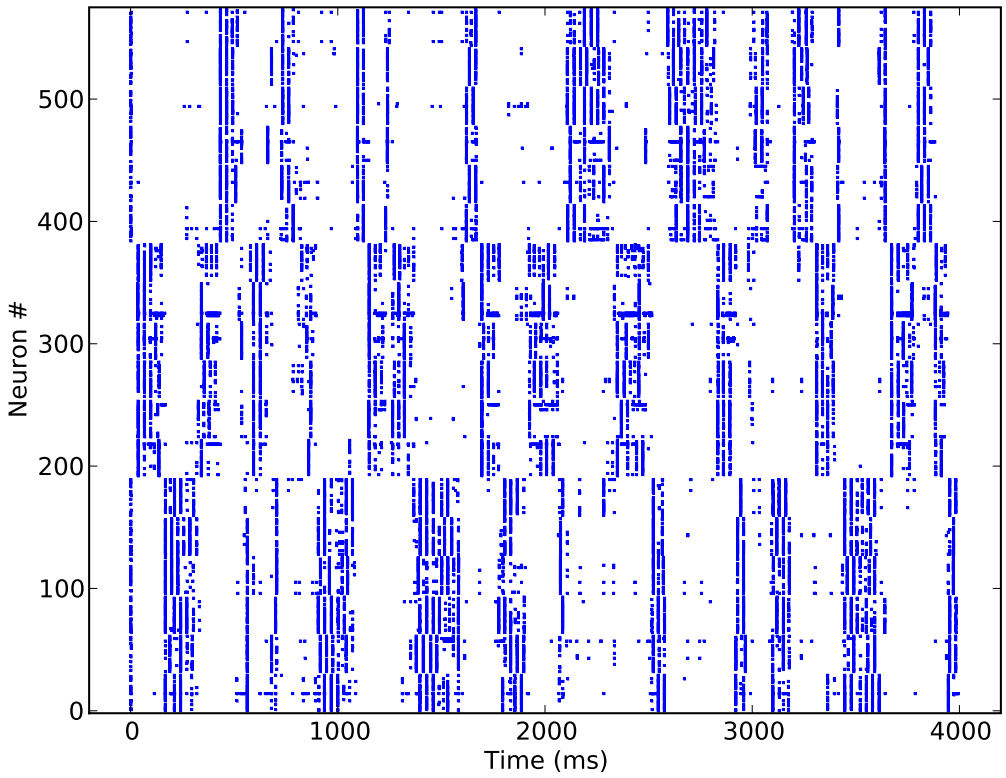


Figure 6.7: Rasterplot of the Layer 2/3 Attractor Memory network simulated on the virtual FACETS Hardware. The network consists of 6 hypercolumns with 3 minicolumns each. One can easily recognize the attractor patterns. .

The Layer 2/3 Attractor Memory Network was run for 4000ms (biological time) on the virtual hardware with the modifications mentioned above. The resulting rasterplot is shown in figure 6.7. Despite all the applied changes to the model, the network clearly exhibits attractor patterns. The time an attractor remains active is a bit smaller than in the default example in section 3.1.1. A statistical analysis of the network states, i.e. the calculation of the mean firing rate and the mean membrane voltage of cells belonging to one attractor has not yet been carried out.
However, qualitative studies with simulations on the virtual hardware have shown, that the Layer 2/3 Attractor Memory network model is rather robust to modifications: attractor patterns could be produced with and without a short-term plasticity mechanism in the synapses, with the default ADEX neuron model instead of leaky IF model, and for different network

sizes (3, 6 or 9 hypercolumns with 3 minicolumns each).
As a next step, the network size should be scaled up to larger networks so that the impact of possible losses of synaptic connections on the wafer can be evaluated. In parallel, the network performance should be analyzed with the methods described in section 3.1.3.

6.4.1.3 Profiling

The simulation of the Layer 2/3 network with 624 neurons for 4000ms on the virtual hardware took on average 1900s for a speedup factor of 10^3 and an integration time step of $\mathrm{d}t = 0.01$ms The same simulation with NEST takes 120ms with time step $\mathrm{d}t = 0.1$ms, which is small enough to produce reliable results. The pure simulation with the executable system specification took 1700s while about 200s were needed for the setup and mapping of the network. Note that the used integration time step used in the simulated neurons doesn't have a severe influence on the simulation time. While the simulation duration can hardly be reduced when a time step of $\mathrm{d}t = 0.1$ms is used, the simulation of the same network with $\mathrm{d}t = 0.001$ms took only 300s longer than with $\mathrm{d}t = 0.01$ms. Thus, most of the computational time of the simulation in the virtual hardware is spent in parts other than the neuron circuits, i.e. for the propagation of spike over the Layer 1 network and in the digital modules for the Layer 2 communication. Hence, the author advises to use $\mathrm{d}t = 0.01$ms as the *default time step* for the numerical integration in the simulated neurons as sometimes the results from a simulation with $\mathrm{d}t = 0.1$ms were unusable.

6.4.2 Synfire Chain with Feed-Forward Inhibition

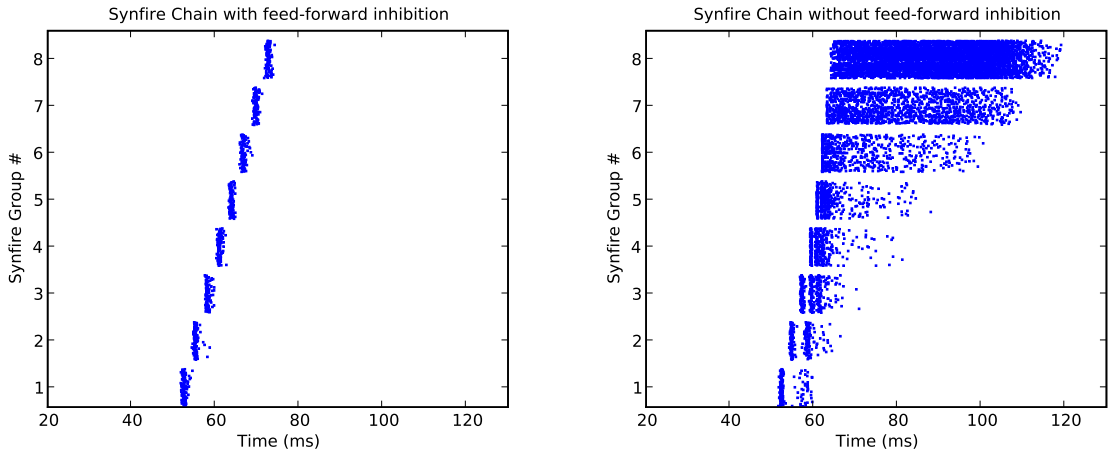


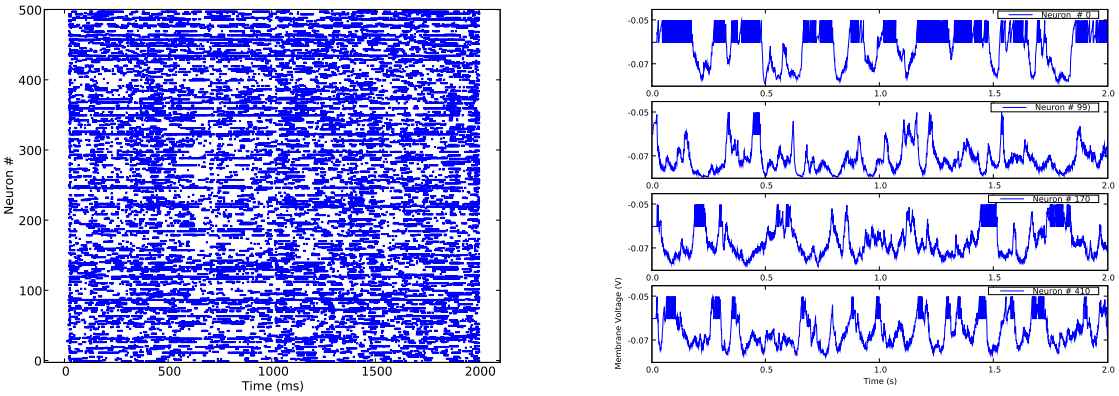
Figure 6.8: Synfire Chain simulated on the virtual hardware. In the left panel: Synfire Chain *with* feed-forward inhibition, inhibitory cells are not shown. The pulse packet properly propagates from one group to the following. In the right panel: the same Synfire Chain, but this time without feed-forward inhibition. Neurons spike several times and the width of the pulse packet diverges.

As the FACETS wafer scale system does not support settable delays for synaptic connections on the wafer, and a routing of delayed synaptic connections over Layer 2 is not yet included into the MappingTool, all synaptic delays are set to *zero*. This seemed critical, as the *Synfire Chain* (see section 3.2) makes extensive use of synaptic delays, especially for the

connection between the inhibitory sub-group to the excitatory sub-group of the same *Synfire Group*. The delayed inhibitory input (4 ms) impedes the excitatory neurons from spiking more than one single time.
When running the *synfire chain* with the parameters from the default model, the synfire chain dies out after the second group. By reducing the synaptic weight of the connections between the inhibitory and the excitatory sub-group, i.e. reducing the strength of the inhibition mechanism, the synfire chain works as it should: pulse-packets properly propagate from one synfire group to the next. An example with 8 synfire groups with and without feed-forward inhibition is shown in figure 6.8. In this run, excitatory neurons spiked only once and the width σ of the pulse packet did not spread.

Some simulations of larger networks were also attempted on the limited setup with 32 HICANNs with 64 neurons each, thus a maximum of 2048 neurons: Stable synfire chains have been realized for a length of up to 10 groups (1250 neurons). For larger networks some synapses were lost so that the synfire chain died out after several groups. This happens although the *routing program* (see section 2.2.3) reports that all synapses have been successfully realized. This error is already noticed during the *parameter transformation*, long before the configuration of the virtual hardware. It seems that the error occurs when the routing information is read out from the HardwareGraph. This presents a solvable technical problem, so the simulation of larger synfire chains should be possible. Also, a statistical analysis of the synfire chain as it is described in section 3.2 should be carried out.

6.4.3 Self-Sustaining Cortical Activity with Asynchronous Irregular Firing Patterns



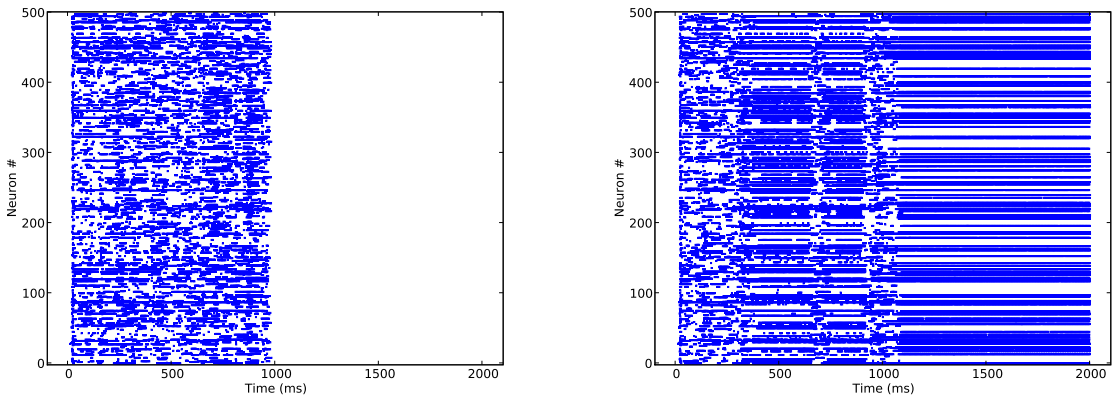
(a) **(b)**

Figure 6.9: Self-sustained Activity with AI states. (a) Rasterplot of the randomly connected network with 500 neurons. Inhibitory neurons have indices ranging from 400 to 500. The activity is sustained for 2000ms. (b) Time course of the membrane voltage of 4 neurons is shown. One can recognize the irregular activity.

The network model showing self-sustained activity with asynchronous irregular (AI) states according to section 3.3 was only tentatively tested with the virtual FACETS Stage 2 Hardware. The randomly connected network consists of 500 cells (80% excitatory / 20% in-

hibitory), 5% of the excitatory neurons are of a low threshold spiking (LTS) type. For all cells, the ADEX neuron model is used. 10% of the network are initially stimulated for 50ms by external spike sources, then the network is left to itself.

Unfortunately, a stable network setup, which is able to reliably produce self sustaining activity with AI states, has not yet been found for the virtual hardware. However, a network configuration that sometimes shows the desired behavior is found by reducing the spike frequency adaptation for the excitatory cells. However, the network showed different behavior in different runs (i.e. when different random seeds were used). An example of sustained AI state activity is shown in figure 6.9. The self-sustained activity has been recorded for 2000 seconds, the rasterplot is shown on the left. The right plot represents the voltage traces from 4 neurons of the network: one can easily recognize the irregularity in their spiking activity. The coefficient of variation of the inter-spike intervals in this case is $\mathrm{CV}_{\mathrm{ISI}} = 2.11$ and the average cross correlation between two arbitrary neurons is $\mathrm{CC} = 0.009 \pm 0.065$, thus the network is in an *asynchronous irregular* state. However, for the same network setup it may also happen



(a) **(b)**

Figure 6.10: Rasterplots for two given examples, when the randomly connected network does not show self-sustained activity with AI states. In (a) the activity of the network dies out, in (b) the network makes a transition into a highly correlated regular state.

that the activity of the network is not sustained or that the networks fires highly correlated and regularly (see figure 6.10). A thorough examination of these phenomena is planned for the near future.

At this point, it has been shown that self-sustaining AI states are possible on the virtual hardware, but still a lot of analysis and parameter optimization have to be carried out for this benchmark model to work properly on the hardware substrate.

7 Achievements and Outlook

7.1 Achievements

Gaining control of a highly complex neuromorphic hardware system is *per se* a difficult task, as one has to deal with an extensive range of hardware imperfections which result in an unpredictable chip behavior. This task gets even more complicated for a large-scale system like the FACETS wafer-scale hardware, which aims to be flexible enough to serve as a universal neuroscientific modeling tool. For this purpose a correspondingly complex, custom-built software framework for operating the neuromorphic device is indispensable.
As the physical hardware is not yet available, an executable system specification serves as the test bench for this support software.

Therefore, the existing system simulation has been modified with focus on functionality and simulation speed. As a result of this procedure the ANNCORE now has its adequate behavioral representation in the virtual hardware, which provides the same configuration space for all functional units as in the real HICANN chip.
An event-based adaptive exponential integrate-and-fire neuron model has been successfully embedded into the simulated ANNCORE of the system simulation along with an analytically exact integrate-and-fire neuron model. The functionality of both was proven by comparing them to analogous software simulations. This endeavour was significantly facilitated by the use of the meta-language PyNN for the specification of the corresponding experiments.
The HICANN implementation of the short-term synaptic plasticity mechanism was accurately reproduced in the virtual hardware. Additionally, a parameter transformation from the original STP model to a configuration of the hardware was developed and also included into the *MappingTool*. Therewith, both the parameter transformation and the virtual STP mechanism could be tested, revealing the impact of the restricted configurability of the hardware. Although it was possible to reproduce synaptic response patterns from software simulations, there is still potential for an enhancement.
One major achievement that was inherent with the further development of the executable system specification and that is not further mentioned in this thesis, is the *constant feedback* that was given to the developers and maintainers of the software stack for the operation of the FACETS Hardware. Not only were these software layers *tested* when operating the virtual hardware, but the enhanced system simulation also gave rise to the implementation of *new features*, such as a hardware specific modification in the *pyNN.hardware.stage2* module or the separate placing of excitatory and inhibitory neurons in the *MappingTool*.

The dynamics of two FACETS Demonstrator Benchmark models could be reproduced on the virtual hardware. This entails two major achievements, the first concerns the operation workflow, the second relates to the performance of the network models:
The software for the operation of the FACETS wafer-scale system has reached a mature state: it is capable to transform a network model described in PyNN into a representation on the hardware and later run the simulation on the virtual hardware. Also, the configuration of the system and the readout of spike data work appropriately.

After applying only minor modifications to the original model, simulations on the virtual hardware of the Layer 2/3 Attractor Memory network exhibit the same attractor patterns as corresponding software simulations. The network proved to be very stable against changes of both neuron and synapse models.
An unexpectedly positive result was the successful implementation of a *synfire chain* on the virtual hardware: even though the wafer-scale hardware does not support any adjustable delays for synaptic connections, a configuration for a stable propagation of pulse packets was found for the executable system specification. In addition to that, the discussions regarding this benchmark model revealed another workaround possibility, namely the implementation of adjustable synaptic delays over the Layer 2 communication network.
The third FACETS Demonstrator benchmark underwent only preliminary testing, nevertheless a network showing self-sustained activity with asynchronous irregular states could be realized, although still lacking functional reliability. Locating the source of this flaw remains a task for the future.
One major constraint of the FACETS system was discovered: the limited bandwidth for sending and recording spikes over Layer 2. To tackle this problem, all simulation of larger networks had to be conducted with a speedup of 10^3 (instead of the default value of 10^4).

However, the virtual hardware was shown to be capable of simulating small biologically relevant models with very different architectures. As of now, scaled versions of the FACETS Demonstrator benchmark models can be simulated with the virtual system in order to further study hardware-specific limitations, such as the loss of synaptic connections due to the limited Layer 1 communication bandwidth.

7.2 Outlook

The FACETS wafer-scale hardware system aims to serve as a universal and highly flexible tool for neuroscientific modeling. Ideally, a modeler should be able to make use of all the benefits of a neuromorphic system (e.g. the speedup or the low power consumption) without considering any technical details or restrictions. The ideal neuromorphic hardware would be as easy to use as any other simulator supported by PyNN. Problems which arise from the complexity of the system, such as synapse loss or discretization of parameters, should be automatically compensated. Neurons should be arbitrarily excitable and recordable. This requires not only a highly configurable hardware system but also a very sophisticated software layer that deals with all these problems.

As any groundbreaking idea still in its infancy stages, the FACETS wafer-scale hardware system still has a long way to go before achieving this dream.
However, the way that is being paved by the FACETS Demonstrator seems to point towards the right direction. In the course of this sub-project, the development of the software layers was substantially advanced, while the level of detail of the virtual hardware was increased. Some emerging problems that limit the performance of the current system are mentioned in the following, together with some suggestions for their solution.
A limited number of individually adjustable spike sources with high stimulation rates appeared as a major constraint when emulating neural network models on the virtual hardware. Therefore, all simulations had to be conducted with a relatively low acceleration factor of only 10^3. To fix this problem, one should implement the on-wafer Poisson spike sources into the virtual hardware (there are 8 per HICANN) and include them also into the Mapping strategy.

In many cases it should be enough to combine on-chip spike sources for neurons to receive a relatively uncorrelated input. This would decrease the need of Layer 2 resources so that a simulation could be run with a higher acceleration factor with the same synaptic input. Another strategy would be to use one part of the neurons on the wafer just as Poisson-like spike sources for the actual experiment.
Regarding the number of neurons which can be recorded during a single run, the user is often limited by the output bandwidth to recording only a fraction of them. If this is not sufficient for subsequent analysis of the network state, an automated procedure should be provided, which runs the same experiment several times and later combines the recordings from different neurons in different runs to produce one single result.

Furthermore, in order to exploit all features implemented in the ANNCORE, the Mapping and Routing strategies have to be tuned. This is especially needed for the usability of the short-term plasticity mechanism, as some STP parameters can only be set for a quarter of synapses per ANNCORE. Neuron circuits in the FACETS Hardware that are switched together allow a very versatile configuration, for example many different time constants and reversal potentials in the synaptic input circuits. Following this approach, also neuron models with various adaptation mechanisms can be implemented. In this respect, the hardware neurons outplay many software simulators that do not yet support these features.

A further refinement of the executable system specification targeting these detailed improvements should definitely be conducted. On the one hand, this will encourage a further enhancement of the software layers responsible for the operation of the neuromorphic hardware system. On the other hand this might reveal bottlenecks and flaws in hardware design, possibly leading to the development of superior modules to be implemented in future versions. A persistent effort in both software engineering and hardware design remains necessary for achieving the declared goal of the FACETS neuromorphic engineering division: to create a superior, universal modeling tool.

Bibliography

Abeles, M., *Local Cortical Circuits: An Electrophysiological study*, Springer, Berlin, 1982.

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

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

Brette, R., Exact simulation of integrate-and-fire models with synaptic conductances, *Neural Computation*, *18*, 2004–2027, 2006.

Brette, R., and W. Gerstner, Adaptive exponential integrate-and-fire model as an effective description of neuronal activity, *J. Neurophysiol.*, *94*, 3637 – 3642, 2005.

Brette, R., and D. Goodman, Brian, 2008, a simulator for spiking neural networks based on Python.

Brette, R., et al., Simulation of networks of spiking neurons: A review of tools and strategies, 2006.

Brüderle, D., Neuroscientific modeling with a mixed-signal VLSI hardware system, Ph.D. thesis, 2009.

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

Davison, A. P., 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*, 2008.

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

Destexhe, A., Self-sustained asynchronous irregular states and up–down states in thalamic, cortical and thalamocortical networks of nonlinear integrate-and-fire neurons, *Journal of Computational Neuroscience*, *27*, 493–506, 2009.

Diesmann, M., M.-O. Gewaltig, and A. Aertsen, Stable propagation of synchronous spiking in cortical neural networks, *Nature*, *402*, 529–533, 1999.

EPFL, and IBM, Blue brain project, 2008.

FACETS, Fast Analog Computing with Emergent Transient States – project website, `http://www.facets-project.org`, 2009.

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

Gewaltig, M.-O., and M. Diesmann, NEST (NEural Simulation Tool), *Scholarpedia*, *2*, 1430, 2007.

Groetker, T., S. Liao, G. Martin, and S. Swan, *System Design with SystemC*, Kluwer Academic Publisher, 2002.

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

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

Indiveri, G., Neuromorphic VLSI models of selective attention: From single chip vision sensors to multi-chip systems, *Sensors*, *8*, 5352–5375, 2008.

Indiveri, G., E. Chicca, and R. Douglas, A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity, *IEEE Transactions on Neural Networks*, *17*, 211–221, 2006.

Kremkow, J., L. Perrinet, A. Aertsen, and G. Masson, Functional consequences of correlated excitatory and inhibitory conductances, *Journal of Computational Neuroscience*, submitted 2009.

Kubota, S., and T. Kitajima, Biophysical model of spike-timing-dependent plasticity based on temporal factors of intracellular Ca2+, *International Congress Series*, *1301*, 127–131, 2007.

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

Lundqvist, M., M. Rehn, M. Djurfeldt, and A. Lansner, Attractor dynamics in a modular network model, *Network: Computation in Neural Systems*, *17*, 253–276, 2006.

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

Matsumura, M., T. Cope, and E. E. Fetz, Sustained excitatory input to motor cortex neurons in awake animals revealed by intracellular recording of membrane potentials, *Experimental Brain Research*, *70*, 463–469, 1988.

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

Mehring, C., 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*, 1253–1254, 2003.

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

Miikkulainen, R., J. Bednar, Y. Choe, and J. Sirosh, *Computational Maps in the Visual Cortex*, Springer, 2005.

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

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

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

Muller, E., L. Buesing, J. Schemmel, and K. Meier, Spike-frequency adapting neural ensembles: Beyond mean adaptation and renewal theories, *Neural Computation*, *19*, 2958–3010, 2007.

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

Petrovici, M., V. Petkov, J. Schemmel, and K. Meier, Exploring the parameter space of a cortical attractor network model, to be published 2010.

Renaud, S., J. Tomas, Y. Bornat, A. Daouzli, and S. Saighi, 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.

Schemmel, J., WP7 STDP implementation, 2006, university of Heidelberg.

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

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

Schneggenburger, R., J. Lopez-Barneo, and A. Konnerth, Excitatory and inhibitory synaptic currents and receptors in rat medial septal neurones, *Journal of Physiology*, *445*, 261–267, 1992.

Shepherd, G. M. (Ed.), *The Synaptic Organization of the Brain*, 5 ed., Oxford University Press, 198 Madison Avenue, New York, New York, 2004.

Shouval, H. Z., G. C. Castellani, B. S. Blais, L. C. Yeung, and L. N. Cooper, Converging evidence for a simplified biophysical model of synaptic plasticity, *Biol. Cybern.*, *87*, 383–391, 2002.

Shu, Y., A. Hasenstaub, and D. A. McCormick, Turning on and off recurrent balanced cortical activity, *Nature*, *423*, 288–293, 2003.

Touboul, J., and R. Brette, Dynamics and bifurcations of the adaptive exponential integrate-and-fire model, *Biological Cybernetics*, *99*, 319–334, 2008.

Varela, J., S. Song, G. Turrigiano, and S. Nelson, Differential depression at excitatory and inhibitory synapses in visual cortex, *Journal of Neuroscience*, *19*, 4293–4304, 1999.

Vogels, T. P., and L. F. Abbott, Signal propagation and logic gating in networks of integrate-and-fire neurons, *J Neurosci*, *25*, 10,786–95, 2005.

Vogelstein, R. J., U. Mallik, E. Culurciello, and R. E.-C. Gert Cauwenberghs, A multichip neuromorphic system for spike-based visual information processing, *Neural Computation*, *19*, 2281–2300, 2007.

Wendt, K., M. Ehrlich, C. Mayr, and R. Schüffny, Abbildung komplexer, pulsierender, neuronaler netzwerke auf spezielle neuronale VLSI hardware, in *DASS'07: Proceedings of Dresdener Arbeitstagung Schaltungs- und Systementwurf*, pp. 127–132 (german), 2007.

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

Acknowledgments

I wish to express my appreciation and gratitude to all persons who contributed in any way to this thesis, be it by contentual, technical, substantial or mental support.

Prof. Dr. Karlheinz Meier for having given me the opportunity to work in this field of research and his unwavering commitment to the FACETS project.

Dr. Johannes Schemmel for an amazing hardare design, for his invaluable advice in any critical situation and for being such an inspiring group leader.

Prof. Dr. Rene Schüffny for the second revision on this work.

All members of the Electronic Vision(s) group for an incredible work atmosphere, a fantastic christmas party and general solidarity.

All Hardies for being always available for questions regarding the hardware and of course for the hardware itself.

All Softies for the craziest super market I have ever lived in, extensive proof reading, endless table-top soccer games and video sessions.

Karsten Wendt, Johannes Partzsch and Matthias Ehrlich from Dresden.

Sebastian Jeltsch for being always open.

Moritz Schilling for being so genuine.

Eric Müller for being the other *heart* of the softies.

Dr. Daniel Brüderle for being the *heart* of the softies.

Bernhard Kaplan for his voluntary supervision at the beginning of this thesis and the teamwork later on.

Mihai Petrovici for his general awesomeness.

My beloved and my family.