Dissertation

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics
of the
Ruperto-Carola University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

Presented by
Dipl.-Phys. Christoph Koke
born in Bielefeld, Germany

Date of oral examination: February 8, 2017

Device Variability in Synapses of Neuromorphic Circuits

Referees: Prof. Dr. Karlheinz Meier
Prof. Dr. Michael Hausmann

Abstract

In this thesis, we worked with the HICANN neuromorphic chip, which is part of the BrainScaleS System. Its analog circuits underlie time-independent variations that occur during the manufacturing process; this effect is known as mismatch. It causes the analog neuron and its synapses to deviate from desired behavior. Each neuron in HICANN has 23 individual parameters, that allow to compensate those variations. We therefore can calibrate the neuron. The variation of the obtained parameters from trial to trial are significant and set the limit for the reachable precision. In this work we develop new calibration methods for the synaptic and membrane time-constants of the neuron and analyze the behavior of the synapse circuits using transistor-level simulations. We analyze the post-synaptic potential (PSP) curves to extract the time-constants by fitting a parameterized model of the PSP recorded from the neurons. The resulting time-constant calibration improves the precision by which the model parameters can be set towards the mentioned limit. Monte Carlo transistor-level simulations are used to show that the relative deviation of the synaptic strength is below 10 % for most of its parameter range. The presented calibration vastly improves the usage of the analog neuron as a leaky integrate and fire model.

Zusammenfassung

In dieser Arbeit haben wir mit dem HICANN gearbeitet, einem neuromorphen Chip, der Teil des BrainScaleS Systems ist. Seine analogen Schaltungen unterliegen zeitunabhängigen Variationen, die während des Herstellungsprozesses auftreten. Dadurch weicht das analoge Neuron und seine Synapsen vom gewünschten Verhalten ab. Jedes Neuron in HICANN hat 23 einzelne Parameter mit denen diese Variationen kompensiert werden können. Dardurch lassen sich die Neuronen kalibrieren. Allerdings unterliegen die Parameter von Versuch zu Versuch signifikanten Abweichungen, die eine Grenze für die erreichbare Genauigkeit festlegen. In dieser Arbeit entwickeln wir neue Verfahren, um die Zeitkonstanten der Synapsen und der Membran des Neurons zu kalibrieren und untersuchen das Verhalten der Synapsenschaltkreise mit Simulationen auf Transistorebenen. Wir analysieren die Kurven der Postsynaptischen Spannungsverläufe (PSP) und extrahieren die Zeitkonstanten, indem wir ein parametrisiertes Modell des PSP an Kurven anpassen, die von den Neuronen aufgenommen wurden. Die resultierende Kalibrierung der Zeitkonstanten verbessert die Genauigkeit, mit der die Parameter gewählt werden können, so dass sie nun fast bis auf die zuvor erwähnte Grenze eingestellt werden können. Wir verwenden außerdem Monte Carlo-Simulationen auf Transistorebene, um zu zeigen, dass die Stärke der Synapsen in weiten Teilen ihres Parameterbereiches eine Abweichung von unter 10 % aufweist. Die vorgestellte Kalibrierung verbessert die Verwendung des analogen Neurons als integrierendes und feuerndes Model mit Leck erheblich.

Contents

1 Introduction

Artificial neural networks are computational models inspired by the nervous system. Such models can help us to gain a better understanding of the function of the brain and also open up new approaches to otherwise computationally complex tasks. Nowadays, more and more tasks that were considered most challenging for classical algorithms are solved by deep neural networks. They can successfully categorize and describe images or beat even professionals in the game of Go [LeCun et al. 2015; Silver et al. 2016].

In addition to such biologically inspired algorithms, there is a large interest in understanding the brain function with the help of computer simulations. A prominent example for this is the BlueBrain Project, which aims to simulate several cortical columns of rats with an unprecedented level of detail [Markram et al. 2015] on the Blue Brain IV supercomputer. However, the Blue Brain IV has a power consumption of 328.70 kW [TOP 500 2015]. Extending these simulations to the scale of the human brain, with its 86×10^9 neurons [Azevedo et al. 2009], appears beyond reach, because neither the computational nor the required electrical power are available.

Thus both current deep neural network implementations as-well-as in-silico neuron experiments are biologically inspired, but their power consumption is orders of magnitude away from that of the human brain itself: only about 20 W. And so is their performance: Even if the brain got beaten in certain tasks, it is still much more versatile. This inspires us to leave classical computers with the von-Neumann architecture behind and design new computational devices to transfer the computational paradigms of neurons into silicon, so called neuromorphic systems. First steps in this direction have been taken with a biologically inspired classifier using spiking neurons [Diamond et al. 2016], for deep neural networks [Indiveri et al. 2015] and with a hybrid architecture which combines the benefits of parallel processing and a learning neural networks with those of shared memory access in traditonal computers [Graves et al. 2016].

The temporal evolution of those artificial neural network models is usually computed in numeric simulations. For networks using spiking point neurons, these have three computational components: First the numerical integration of the neuron model, second the integration of the synaptic events and third the distribution and processing of the spike-based communication. The first and the second tasks may be computationally costly, but can be scaled and distributed in large simulations over many computational cores. However, the third has become the limiting factor in growing network size [Kunkel et al. 2014; Zenke et al. 2014].

Neuromorphic solutions can provide an alternative approach to numerical computations. They can decrease computation time and power consumption of neuronal

simulations and specialized systems can improve upon the spike-based communication. However, while numeric simulations allow us to implement any mathematical model or communication scheme, neuromorphic systems usually have to pay for the increased performance by a loss of generalization or they have to sacrifice precision and deterministic behavior.

In this thesis, we work with such a neuromorphic system, that uses an analog neuron model. In this system each neuron is subject to small variations in the manufacturing process. The aim of this thesis is it to compensate for those variations to regain an uniform neuron behavior. For this task, we first characterize the deviations from the desired behavior and then provide a calibration method, that restores the desired behavior.

Of particular interest are neuromorphic systems that use very-large-scale integration (VLSI) to combine the required features on single microchip because they promise to give the largest improvement in efficiency [Hasler et al. 2013]. Those systems come in a large variety [Furber 2016]. For instance, the SpiNNaker project uses custom processing units, each consisting of 18 ARM cores and a special packet routing unit [Furber et al. 2014]. The ARM cores only provide fixed-point arithmetic and the routing system is designed to send many small packets to many targets. Each unit can transport events to six neighboring cores, which allows the system to scale arbitrarily. Since the routing is not synchronized with the processing units, the system can no longer do deterministic simulations. This approach is still very close to classical simulations. In fact, the SpiNNaker core still uses a von-Neumann architecture and can compute arbitrary neuron models and synapse types, but they drop the costly synchronization process between the singular nodes.

The TrueNorth system by the IBM Synapse project leaves the von-Neumann architecture behind. It has no longer a general purpose computing unit, but implements a fully digital custom neuron and synapse model [Amir et al. 2013; Cassidy et al. 2013; Esser et al. 2013]. Even if the models depend on stochastic input, the system still produces deterministic results. The digital neurons are organized in cores, of which each contains 256 neurons with 256 input synapses each. 4096 cores are integrated in one TrueNorth chip using a 22 nm technology leading to the impressive number of one million neurons per chip. The high integration requires that some parameters are shared between synapses or neurons.

Like SpiNNaker, TrueNorth also provides a custom routing system, that allows the communication from every single neuron to every other neuron. As a performance benchmark, TrueNorth allows real time image recognition on a 400-pixel-by-240-pixel video input at 30 frames per second while the chip is consuming 63 mW [Merolla et al. 2014] and efficient image classification with deep learning networks processing more than 6000 frames per watt [Esser et al. 2016]. These fully digital solutions reduce the power consumption greatly, while remaining quite flexible in the available neuron models.

As an alternative to digital neurons and synapses, there are analog physical neuron models in silico. The neuron and synapse are emulated by analog electrical circuits. These circuits can emulate neurons by transferring their electrical properties into

analog VLSI circuits [Indiveri et al. 2011].

For example Neurogrid implements a custom analog neuron model with shared analog dentrites [Benjamin et al. 2014]. Neurogrid consists of so called Neurocore chips, each containing 65 536 neurons. The neurons operate in real time. The communication is digital and multiple Neurocores are using a tree-like routing structure with address–event representation (AER) [Silver et al. 2007]. Each Neurocore consumes 150 mW [Furber 2016]. To achieve such power efficiency , parameters and synapses are shared between neurons, which reduces the freedom in network design.

A further example is Spikey, which has 384 analog circuits. Opposed to Neurogrid these have partialy individual parameters [Schemmel et al. 2006; Brüderle et al. 2011] and operate above-threshold with an accelerated speed compared to biological networks by a factor of 10^4. This compensates for the higher energy consumption. Further its synapses model provides on-chip Short Term Plasticity (STP) and Spike Timing Dependent Plasticity (STDP). While Spikey is a versatile computing substrate [Pfeil, Grübl, et al. 2013; Pfeil, Scherzer, et al. 2013], the number of neurons per chip is limited.

Our group developed the BrainScaleS System [Schemmel et al. 2010; Brüderle et al. 2011] as a successor of Spikey. The core of the system is the High Input Count Analog Neuronal Network (HICANN) chip, which is the base for this work. A BrainScaleS System has 384 HICANNs. The wafer-scale integration interconnects the HICANNs on the directly on the production wafer. This increases the neuron density and creates a sophisticated bus network for interconnecting the neurons. As outstanding feature, a single system provides 196 608 individually paramamized analog neuronal circuits emulating the Adaptive Exponential Integrate-and-Fire (AdEx) neuron model [Brette et al. 2005]. Each neuron has at least 220 individual analog synapses, which support STP and STDP.

In addition the temporal evolution of the neurons is faster than the one of their biological counterparts. The speed up factor is about 10^4. Compared to systems like Neurogrid, which operate in real-time, this increases the power efficiency of the system in terms of energy per spike [Benjamin et al. 2014].

The VLSI implementation of analog neurons and synapses raises two challenges. Firstly, the performance of an analog neural circuits suffers from mismatch: each instance of any supposed-to-be-identical circuit slightly differs from its specification, due to uncertainties in the manufacturing process [Pelgrom et al. 1989; Razavi 2001]. Secondly, each neuron circuit has 23 individual voltage and current parameters – in total about 12 000 per chip – which have to be generated on the chip and also will underlie variations. The source of these voltages and currents are subject to substantial variations between individual experiments.

Those challenges can be addressed by calibrating the neurons. A first calibration method developed for HICANN has only been applied to neurons without synaptic input [Schwartz 2013]. In this work, we extend the previous work of Schwartz [2013] and present a calibration for the synaptic time-constant of the analog neuron circuits. In addition we show how the trial-to-trial variation is related to the mismatch effect and characterize the variations that arise in the synapses. After application of

the presented calibration routines, the analog neuron circuits are ready for neural network emulations, using a Leaky Integrate-and-Fire (LIF) neuron model.

This thesis comprises two main parts. First we introduce the system and our methods: In Chapter 2 we present the physical hardware of the BrainScaleS System in further detail, while we follow in Chapter 3 with the commissioning of the system. This includes the operational software of the system and tests to verify its functionality. In Chapter 4 we define our calibration task and present our measurement and simulation methods.

In the second half of this work we present our main results: We worked with two revisions of HICANN: the earlier revision 2 and the most recent revision 4.1, which had a major redesign in the synaptic circuits of the neuron. In Chapter 5 we work towards the calibration of the synaptic and membrane time-constants. We show how mismatch affects the programming of the analog parameter storage, repeat the calibration for the basic neuron potentials for the newest HICANN revision and provide a model to describe the strength of spike events in HICANN. In Chapter 6 we work with HICANN's earlier revision 2. We characterize the integration of synaptic events in the neurons. We show that the use of the neurons is limited by mismatch in the neuron circuits, but we can still provide a preliminary calibration for the neuron in this HICANN revision. Our findings lead to the new revision 4.1, in which an additional compensation for those mismatch effects was integrated. In Chapter 7 we present the changes in detail and show the improvements. We present methods to calibrate the synaptic and membrane time-constants of the neurons. With this result we have a fully calibrated conductance based LIF neuron model, that is available for experiments. We then conclude this work and give an outlook on potential continuing research in Chapter 8.

2 BrainScaleS System

In this chapter we present the BrainScaleS System and give here an overview of the system parts relevant for our work. We begin with a short introduction to the neuron model, then describe the HICANN chip and then provide an overview of the complete system. For further details we refer to the Neuromorphic Platform Specification [SP9 Spec 2015] and to publications related to the components at the relevant sections. Many people contributed to the development of the BrainScaleS System in various ways; we list the involved persons in the Acknowledgments.

The underlying neuron model for the VLSI neuron circuits is the AdEx model [Brette et al. 2005; Naud et al. 2008]. We describe this model and a set of solutions required to analyze our results in Section 2.1. Afterwards we describe the HICANN chip in Section 2.2. There we give a broad overview but focus on the neurons and the synaptic input, which are the main subjects of our work. Finally, in Section 2.3 we describe how HICANN is integrated in the BrainScaleS System. This includes the FPGA Communication PCB (FCP) and the Analog Readout Module (AnaRM), which are required to configure the system and to read back analog signals from the neuron circuits.

2.1 Adaptive Exponential Integrate-and-Fire Model

Biological neurons are the highly specialized cells of the nervous system, which can be found in a large variety [Alberts et al. 2007]. In theoretical neurosciences those cells are studie using different kinds of mathematical models [Gerstner et al. 2002]. The models have to find a balance between the level of abstraction and usability. More complex models become harder to treat mathematically and more expensive in simulations.

For the BrainScaleS System, the Adaptive Exponential Integrate-and-Fire (AdEx) model [Brette et al. 2005; Naud et al. 2008] was chosen as archetype of the neuron circuit [Millner 2012; Schemmel et al. 2008]. It is a point neuron model and abstracts as such the complete spatial expansion of the neuron. It extends the LIF model [Dayan et al. 2001], which was already used in HICANN's predecessor Spikey [Schemmel et al. 2007], by an exponential term that models the strong voltage rise on the membrane occurring during action potentials. Further a second differential equation is added to model spike-based and sub-threshold adaptation. The synapses are modeled using exponentially decaying conductance. For the hardware implementation it is necessary to fix the number of synaptic conductances to two: one for excitatory and one for inhibitory synapses. We take this into account in the mathematical description of the model. In general this is a simplification, because

more conductances are possible if hardware neurons are directly interconnected. But in this work we do not make use of that feature.

In total the model has then four state variables: The membrane voltage V, the adaption current I_w, the excitatory conductance g_e and the inhibitory conductance g_i. For a series of stimulating excitatory spikes at times $t_0^e \dots t_k^e$ with individual weights $w_0^e \dots w_k^e$ and inhibitory spikes at times $t_0^i \dots t_l^i$ with individual weights $w_0^i \dots w_l^i$, the dynamics of this model are given by

$$\begin{aligned} C\frac{\mathrm{d}V}{\mathrm{d}t} &= g_\mathrm{l}\left(E_\mathrm{l} - V\right) + g_\mathrm{l}\Delta_T \exp\left(\frac{V - V_{thresh}}{\Delta_\mathrm{T}}\right) \\ &\quad + g_e\left(E_\mathrm{synx} - V\right) + g_i\left(E_\mathrm{syni} - V\right) + I_\mathrm{ext} + I_w, \end{aligned} \tag{2.1}$$

$$\tau_w \frac{\mathrm{d}I_w}{\mathrm{d}t} = a\left(V - E_l\right) - I_w, \tag{2.2}$$

$$\tau_\mathrm{syne}\frac{\mathrm{d}g_e}{\mathrm{d}t} = -g_e + \sum_k w_k^e \delta\left(t_k^e\right) \text{ and} \tag{2.3}$$

$$\tau_\mathrm{syni}\frac{\mathrm{d}g_i}{\mathrm{d}t} = -g_i + \sum_l w_l^i \delta\left(t_l^i\right). \tag{2.4}$$

The exponential term causes a fast blow-up of the solution as soon as V gets close enough to V_thresh. In this case V reaches infinity in a finite time interval. For the numerical solutions and also for the circuit emulation this is not feasible. Therefore, a spike and reset condition like in the LIF model is added, so that if

$$V > V_\mathrm{t} \text{ then } V \to V_\mathrm{reset} \text{ and } I_w \to I_w + b. \tag{2.5}$$

This gives 15 parameters in total, which are listed in Table 2.1. In contrast to the original model, V_reset and V_thresh are added as parameters, because the implementation on HICANN allows to choose them independently. For $a = 0$, $b = 0$ and $\lim_{\Delta_\mathrm{T}\to 0}$, Equation (2.1) converges to a LIF model with conductance based synapses:

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = g_\mathrm{l}\left(E_\mathrm{l} - V\right) + g_e\left(E_\mathrm{synx} - V\right) + g_i\left(E_\mathrm{syni} - V\right) + I_\mathrm{ext}. \tag{2.6}$$

In this work we use the hardware neuron in this form, because it will simplify the analysis of the synaptic and membrane time-constants.

In total, the AdEx model is well suited for analog emulation. It requires only four dynamic variables and can reproduce a large number of activity patterns observed in biological neurons [Naud et al. 2008]. Two examples recorded on HICANN are shown in Figure 2.1.

2.1.1 Solution for the Steady State

In the hardware neuron in HICANN the membrane V is the only state variable, that can be -measured. We therefore need to infer information about the other state

C	membrane capacity
E_l	membrane leakage potential
g_l	leakage conductance
Δ_T	slope factor of the exponential
E_synx	excitatory reversal potential
E_syni	inhibitory reversal potential
τ_w	adaptation time-constant
a	sub threshold adaptation
b	spike-triggered adaptation
V_t	spike threshold
V_reset	reset potential
V_thresh	threshold potential for the exponential term
τ_syne	excitatory synaptic time-constant
τ_syni	inhibitory synaptic time-constant and
I_ext	external current

Table 2.1 Parameters of the AdEx model, for Equations (2.1) to (2.5).

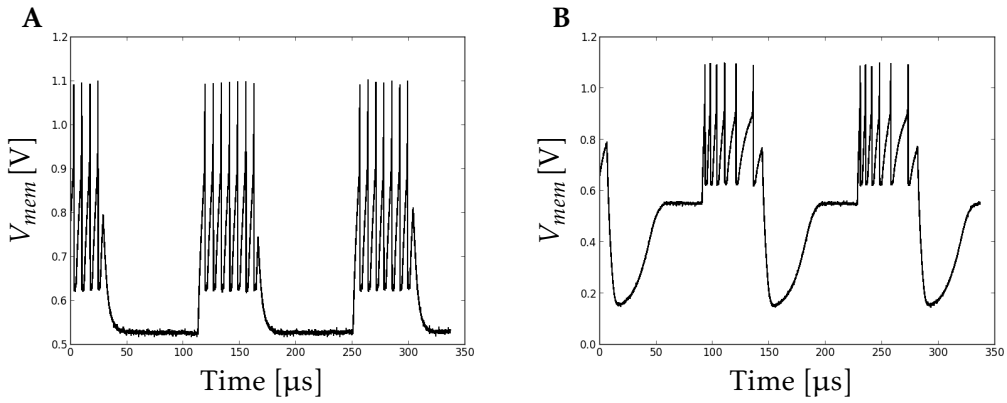


Figure 2.1 Reproduction of firing patterns on analog neurons on HICANN using a current stimulus. The shown pattern were described as **A** Tonic spiking **B** Spike frequency adaptation [Naud et al. 2008]. For tonic spiking the adaptation is disabled and the neuron directly react to the increase current stimulus I_ext with regular spikes. The strong onset caused by the exponential term is well visible. For the spike frequency adaptation b is set larger than 0. This causes the adaptation current I_w to rise after each spike, which slows down the spike rate. After the current pulse ends, the membrane is pulled below the resting potential by the adaptation current I_w. Figures taken from Tran [2013].

variables from the membrane. For this we seek analytic solutions which help us to overcome this deficit.

We often observe the neuron membrane in a steady state, which is the state of the neuron where $\frac{\mathrm{d}V}{\mathrm{d}t} = 0$. In the steady state the neuron stays at an effective resting potential E_l^{eff}, depending on the strength of the conductance,

$$E_l^{\mathrm{eff}} = \frac{g_i \cdot E_{\mathrm{syni}} + g_x \cdot E_{\mathrm{synx}} + g_{\mathrm{l}} \cdot E_l}{g_i + g_x + g_{\mathrm{l}}}. \tag{2.7}$$

2.1.2 Solution for a Single Input Spike

For our calibration task, we need to determine the synaptic time-constants τ_{syne} and τ_{syni} of the neuron, where we are still limited to the membrane potential V as only measurable state variable. We can do this by stimulating the neuron with a single spike and analyze the course of the membrane, the so called Postsynaptic Potential (PSP). Here we derive a mathematical form for a Postsynaptic Potential (PSP) that we can fit to PSPs recorded on HICANN to determine τ_{syn}.

We simplify the task by setting two preconditions: we operate the neuron only as LIF neuron as described by Equation (2.6) and we require that the neuron is in its steady state, when the spike event arrives. This implies that the resulting PSPs do not overlap. Given this conditions a solution for Equations (2.6) to (2.4) can be found [Rudolph et al. 2006]. However for our needs an approximation for single spikes is more useful [Bytschok 2011; Petrovici 2016]. For a spike at t_0 with weight w occurring when the neuron is in the steady state, we then obtain the shape of a single PSP

$$V(t) \approx \begin{cases} E_l + \theta(t_0) \cdot A \cdot \left(\exp\left(\frac{t_0 - t}{\tau_m}\right) - \exp\left(\frac{t_0 - t}{\tau_{\mathrm{syn}}}\right) \right) & \text{if } \tau_m \neq \tau_{\mathrm{syn}} \\ E_l + \theta(t_0) \cdot \frac{w}{\tau_{\mathrm{syn}}} \cdot \exp\left(-\frac{t - t_0}{\tau_{\mathrm{syn}}}\right) \cdot (t - t_0) & \text{if } \tau_m = \tau_{\mathrm{syn}}, \end{cases} \tag{2.8}$$

$$\text{with } A = \frac{w \cdot \left(E_{\mathrm{syn}} - E_{\mathrm{l}}\right) \cdot \tau_g}{g_{\mathrm{l}} \cdot \tau_m}, \tag{2.9}$$

$$\text{and } \tau_g = \left(\frac{1}{\tau_{\mathrm{syn}}} - \frac{1}{\tau_m} \right)^{-1}. \tag{2.10}$$

In this form Equation (2.8) is not yet well suited to be fitted to a recorded PSP. It still contains to many free parameters, which causes problems in a fit because these show a correlated response to changes. Further, we cannot determine good initial parameters for the fit, which are important for a good convergence of the fit.

We can improve this by reformulating Equation (2.8) to use the amplitude of the PSP as parameter. We define it as the maximum of the PSP occurring at $\frac{\mathrm{d}V}{\mathrm{d}t} = 0$ and

call it the height h of the PSP. We first need to obtain its position $t_{\max}$ and obtain

$$t_{\max} = \frac{\log \frac{\tau_{\text{syn}}}{\tau_m} \cdot \tau_{\text{syn}} \cdot \tau_m}{\tau_m - \tau_{\text{syn}}}. \tag{2.11}$$

We then insert the result into Equation (2.8) and get the height h of the PSP as

$$h = V(t_{\max}) = A \cdot \left(\tau^{\frac{\tau}{1-\tau}} - \tau^{\frac{1}{1-\tau}} \right) \quad \text{with} \quad \tau = \frac{\tau_{\text{syn}}}{\tau_m}. \tag{2.12}$$

The same considerations hold for the case where $\tau_m = \tau_{\text{syn}}$. We then replace A in Equation (2.8) and obtain the final result for a single PSP as

$$V(t) \approx \begin{cases} E_l + \theta(t_0) \dfrac{h}{\tau^{\frac{1}{1-\tau}} - \tau^{\frac{\tau}{1-\tau}}} \left(\exp\left(\frac{t_0 - t}{\tau_1}\right) - \exp\left(\frac{t_0 - t}{\tau_2}\right) \right) & \text{if } \tau_1 \neq \tau_2 \\ E_l + h\,\theta(t_0) \exp\left(1 - \frac{t - t_0}{\tau_1}\right) \frac{t - t_0}{\tau_1} & \text{if } \tau_1 = \tau_2, \end{cases} \tag{2.13}$$

$$\text{with } \tau = \frac{\tau_2}{\tau_1}, \tag{2.14}$$

where we replaced τ_{syn} and τ_m by τ_1 and τ_2 because the equation is now invariant in exchanging them. This form is better suited to be fitted, because we could eliminate multiple parameters and we can determine good initial values for the remaining free parameters.

For our measurements we need to reconstruct the information how to assign τ_1 and τ_2 to τ_{syn} and τ_m . We use multiple measurements, where one of the two time-constants is fixed value and the other one varied, to do this.

2.1.3 Parameter Domain of the Model

The model itself does not imbue any restrictions on the parameters, but the physical properties of the neuron circuits in HICANN do. As we see in the later chapters these typically use voltages in the range from 0.3 to 1.3 V and membrane and synaptic time-constants occur in the range of 0.1 to 10 µs. We call these the hardware parameter domain of the model. These are significant different from parameters used in biologically oriented models, which we refer to as biological domain. How these parameters biologically oriented models are converted to match the technical domain is shown in Section 4.1.2. For avoiding confusion we consistently use in this work the hardware domain for model parameters.

2.2 HICANN

The HICANN is the functional core of the BrainScaleS System. It implements a neuron model with 512 individually configurable analog spiking AdEx neurons.

Each neuron has 220 (224 in revision 2) analog synapses. Over 12 000 floating gate cells provide individual configurable voltages and currents to the neurons and other subsystems. Spikes are communicated over an on-chip network as 6 bit packets. Each HICANN has two analog output channels for recording the membrane of the neuron and other sources for debugging. A picture of HICANN with its components highlighted is shown in Figure 2.2. The design gives the chip a horizontally mirrored symmetry, to that we will refer to as top and bottom half of the chip.

The global clock of HICANN is generated by a Phase-Locked Loop (PLL) and can be set to values up to 250 MHz. For regular usage a frequency of 100 and 125 MHz has been proven to be stable [Kononov 2011].

The HICANN is either produced as single chip or as whole wafer with wafer-scale integration. The wafer-scale integration is described in Section 2.3.

In total, five revisions of HICANN were developed and four of them produced. In 2009 revision 1 was produced as single chip only for the first HICANN tests and is no longer in use. It was followed in 2011 by revision 2 which was first produced as single chip for tests and then as whole wafer with wafer-scale integration. In 2011 the planning and design of revision 3 was started by Sebastian Millner, but dropped in favor of the Multi-Compartment Chip (MCC) [Hartel 2016; Millner 2012]. In 2015 our results shown in Chapter 6 led to revision 4, which has an improved synaptic input. As we show in Chapter 7, the improvements were substantial. For the wafer-scale integration of revision 4 an additional change was applied and the HICANN is therefore labeled with Revision 4.1. Most of the BrainScaleS Systems use currently revision 2, but the systems will be updated to revision 4.1 step by step.

We now give an overview of the most important HICANN components, the analog neuron, synapses, spike communication and floating-gate parameter storage. If not noted otherwise the description is valid for all revisions. For the notations of voltages and currents in the circuits we primarily follow the names given in the specification or in previous publications. An overview of all used names is given in Appendix C.

2.2.1 Neuron Circuits

The differential equations need to be transferred into analog circuits that emulate them [Millner 2012; Millner et al. 2010]. Physically the neurons are placed in two rows each containing 256 neurons on the HICANN. This is well visible in Figure 2.2. Each neuron has two conductance-based synaptic input circuits, which integrate the spikes coming from the 220 (224 in revision 2) synapses connected to them. Further neurons can be short-circuited to enlarge the number of possible synapses at the cost of reducing the total number of available neurons. The currents and voltages for the neuron are provided by the analog parameter storage individually for each neuron. The original description of the HICANN neuron can be found in [Millner 2012].

For this work only the components that form a LIF neuron are relevant. Its parts are shown in Figure 2.3. The membrane capacity C is a capacitor with nominal

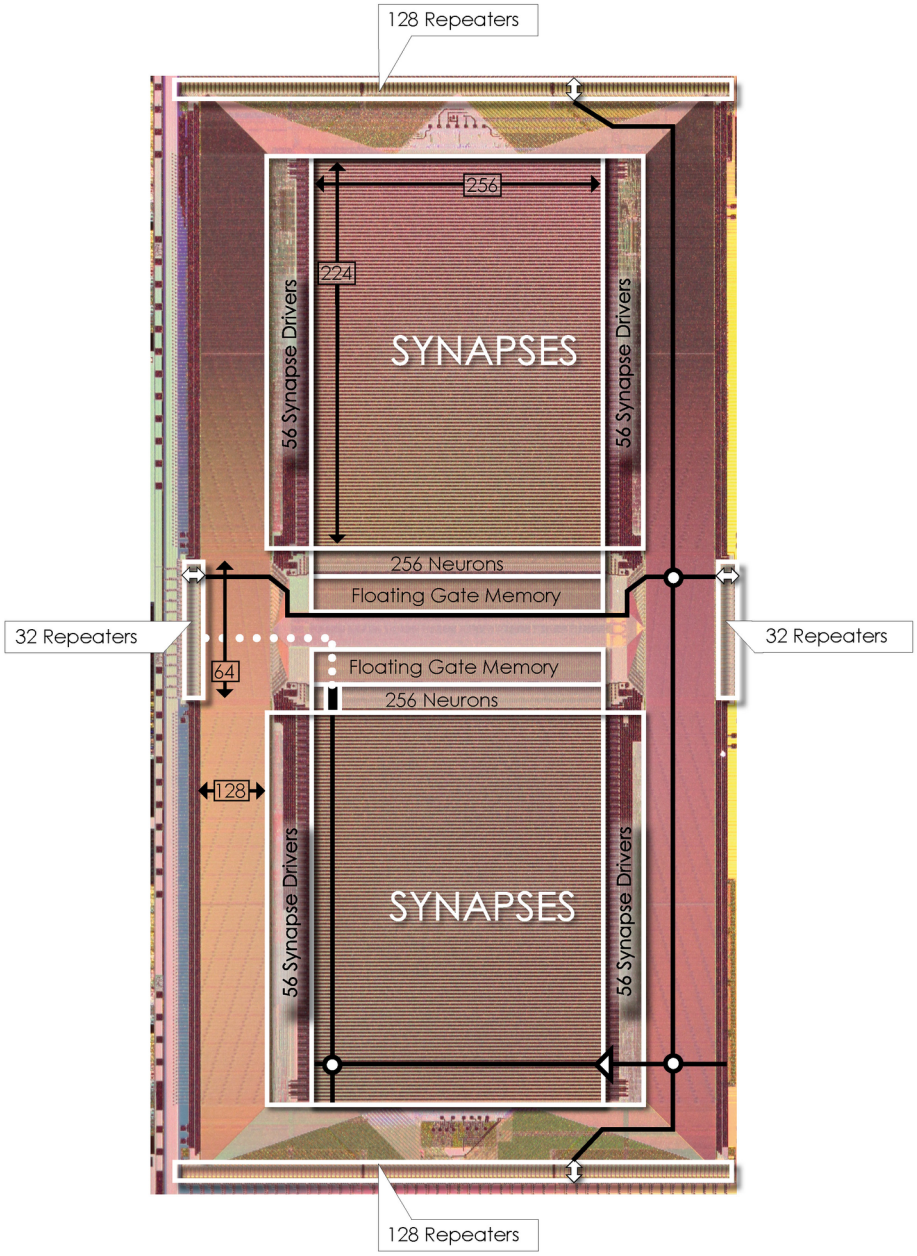


Figure 2.2 Photograph of the HICANN revision 2 without postprocessing for wafer-scale integration. The chip is 5×10mm. Its vertical symmetry is well-defined. On each half are 256 interconnectable neruon circuits with each 224 synapses. The outer parts of chip contain the on-chip routing network. Picture taken by Dan Husmann, annotations by Elisa Ziegenbein.

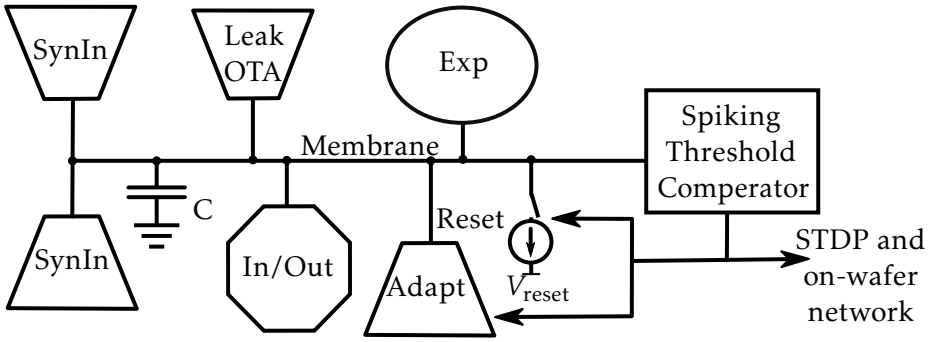


Figure 2.3 Reduced schematic of the HICANN neurons. All components are interconnected by the membrane.

160 fF, where we denote the design goal for the given quantity as nominal values. This value is subject to variation caused by production and especially for capacitors, it is inevitably increased by the parasitic capacitance, caused by other circuit elements. In the neuron circuit the parasitic capacitance cannot be neglected, as it is about 150 fF [Schmidt 2014]. Additionally, a second capacity with nominal 2000 fF can be connected to emulate slower time-constants.

The reset mechanism pulls the neuron towards the reset potential V_{reset} as soon as the membrane voltage V_{mem} crosses spike threshold V_{t}. It can be disabled by a digital parameter, but this is only intended to be used when interconnecting neurons. The current used for the reset is strong enough to overpower any other circuit in the neuron. Further, a spike event can be emitted to the on-chip or on-wafer network. This and the address of the event are controlled by further digital parameters. Then the neuron membrane is hold at V_{reset} for the length of refractory period t_{ref}. Its duration is controlled by the parameter I_{pl}.

The leakage term of Equation (2.1) is built by using an operational transconductance amplifier (OTA) in negative feedback. The OTA creates a current I_{gl} pulling back the membrane towards resting potential E_{l}. For an ideal OTA the current is

$$I_{\text{gl}} = g_{\text{l}} \cdot (E_{\text{l}} - V_{\text{mem}}), \tag{2.15}$$

which perfectly mimics the leakage term of Equation (2.1). The gain g_{l} of the OTA is also the leakage conductance of the neuron model. It is controlled by the current parameter I_{gl}. The actual characteristic of the OTA are shown in Figure 2.4. The linear range of the OTA is limited to a small range of about 100 mV around E_{l}.

Input spikes are handled by the two synaptic inputs of the neurons. Which are different in revision 2 and 4/4.1. A schematic of the input in revision 2 is shown

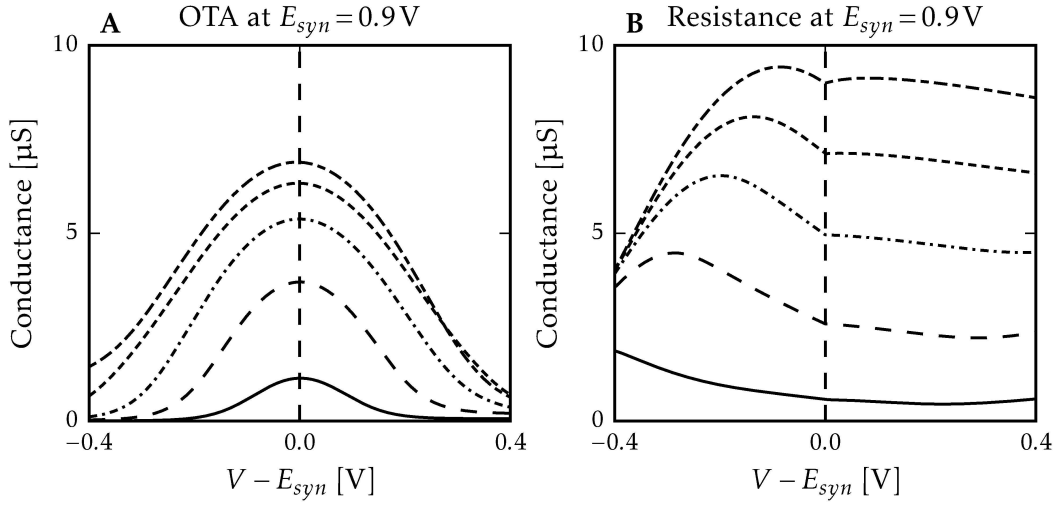


Figure 2.4 Circuit simulation of the conductance through the OTA (**A**) and the resistive element (**B**) in dependence of the difference between reversal and membrane potential for a fixed reversal potential of 0.9 V. The different lines result from different bias currents for the circuit beginning from the bottom 0.1, 0.2, 0.5, 1.0 and 1.5 µA.

in Figure 2.5. These work in two stages: The first stage processes the current pulses from the synapses, which are characterized by their length t_{syn} and strength I_{syn}. An operational amplifier (OP) integrates the current pulses and increases the voltage $V_{\mathrm{integrator}}$ in an RC-circuit. The integrated pulses exponentially decay in the RC-circuit towards the reference voltage V_{syn}, as described by Equations (2.3) and (2.4). In revision 2 the capacity is $C = 249\,\mathrm{fF}$ and was changed in revision 4/4.1 to $C = 472\,\mathrm{fF}$. The synaptic time-constant is controlled by the voltage controlled resistor in the RC-circuit via the voltage parameter V_{syntc}, these are different in revisions 2 and 4/4.1.

The second stage controls the synaptic conductance. In revision 2 the voltage difference between $V_{\mathrm{integrator}}$ and V_{syn} is translated by OTA_1 into a bias current for OTA_0, which then generates the synaptic current $I_{\mathrm{syne/i}}$ flowing onto the membrane. For ideal OTAs it is

$$I_{\mathrm{syne/i}} = g_{e/i} \cdot (E_{\mathrm{syne/i}} - V_{\mathrm{mem}}) \quad \text{and} \tag{2.16}$$

$$g_{i/x} \propto g_{\mathrm{conv}} \cdot (V_{\mathrm{syn}} - V_{\mathrm{integrator}}), \tag{2.17}$$

where g_{conv} is controlled by the current parameter I_{conv}. For revision 2 both inputs are identical.

In revision 4/4.1, OTA_0 was replaced by a current controlled resistor, which has a more stable conductance for larger voltage differences. The characteristics of the OTA and the resistor are shown for comparison in Figure 2.4. The two inputs

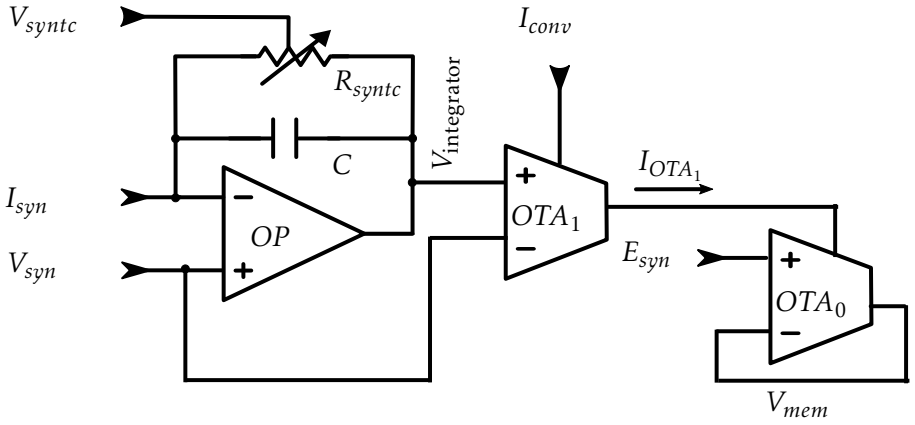


Figure 2.5 Schematic of the synaptic input of HICANN revision 2.

are no longer identical, because the resistor is placed with a different polarity in both inputs. Further a bias generator was added to correct for mismatch in the OP and OTA of the integrator. A schematic of the input for revision 4/4.1 is shown in Figure 2.6. The functional implication of the changes between revision 2 and 4/4.1 are discussed in Section 7.1

To emulate the complete AdEx model, the neuron also has circuits for adaptation and the exponential term. Because we want to use the neuron as LIF neuron, we do

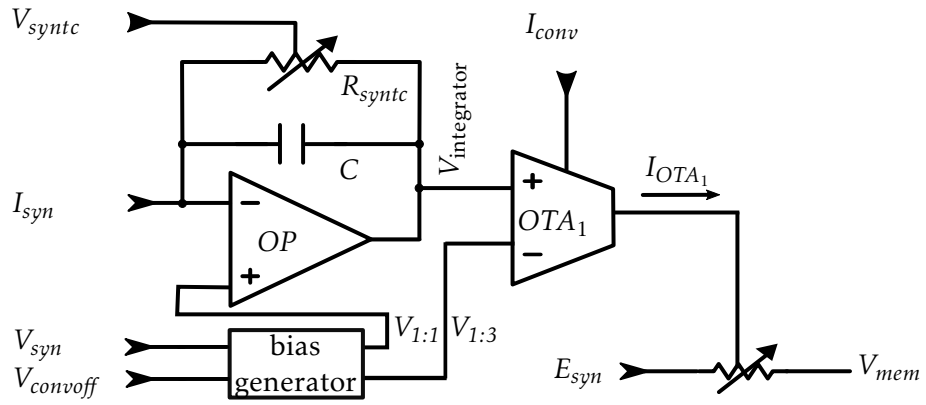


Figure 2.6 Schematic of the synaptic input of HICANN revision 4/4.1. Compared to revision 2, the capacity is enlarged, the resistor in the RC-circuit was exchanged, the reference voltage for OTA_1 can now be shifted by the bias generator and OTA_0 is replaced by an current controlled resistor.

not require these features and have to take care that these circuits do not influence our measurements [Schmidt 2014; Friedrich 2015; Kiene 2014].

2.2.2 On-Chip Spike Communication

A further essential feature of HICANN is the on-chip spike routing, that allows creating arbitrary connections between neurons [Schemmel et al. 2008; Schemmel et al. 2010]. We give here a brief introduction of the on-chip spike communication, a detailed description can be found in the PhD-thesis of Jeltsch [2014] and the project specification [SP9 Spec 2015]. The backbone of the on-chip spike communication are the 128 vertical and 64 horizontal buses on each HICANN. The horizontal and vertical buses can be interconnected by sparse switch matrices. In the wafer-scale integrated HICANN, as described in Section 2.3, repeater circuits extend the buses over the borders between HICANNs. Repeater circuits amplify the signal crossing the borders, so that they can be transferred wafer-wide.

Spike events are transmitted using asynchronous transmission. Each repeater or synapse driver uses a Delay-Locked Loop (DLL) to recover the generating clock of the signal for addresses decoding. This process is also referred to as repeater locking. It requires some time, which is not yet measured [Ziegler 2013]. The spikes are transmitted to the synapses via 224 synapse drivers. Each driver distributes the spike events into two lines of 256 synapses each, which are connected to the neurons on the top or on the bottom of the chip. Furthermore, each synapse driver is sparsely connected to vertical buses on its own and the left or right neighboring HICANN. Per HICANN 8 repeaters of horizontal buses can be used to receive either external spike input, or the spikes of the neurons.

The on-chip network transmits 8 bit packets, which encode the sending neuron in 6 bit and a start and stop bit. In total, every 16 bit one spike can be transmitted because two packets cannot follow directly on each other, The transmission delay depends on the number of repeaters to be passed and cannot be delayed on-chip by other means. Each repeater in the connection will increase the delay further. The repeaters can send a packet every two clock cycles of the PLL. For a PLL of 100 MHz a single bus has a maximum spike rate of 50 MHz. The encoding allows merging spikes of up to 64 neurons onto one bus. The identifier can be configured freely for each neuron individually and is then used by the synapse drivers and synapses to filter the events for the designated neurons.

An addition, to the neurons 8 background generators can merge events onto the buses. They can produce either equidistant spikes or poisson-like distributed spike trains, both with a selectable target rate. We will use these in experiments as a source of reliable equidistant spikes.

2.2.3 Synapses

The synapses are organized in two arrays, one on the top half and one on the bottom half of the chip [Schemmel et al. 2008]. This setup is well visible in Figure 2.2. Each

array can only give input to neurons on its side of the chip. Interconnecting neurons from the top and the bottom can help to overcome this limitation.

Each array has 224 rows and 256 columns, one column per neuron. That gives 114688 synapses on each HICANN revision 2. On HICANN revision 4/4.1 four synapse lines needed to be removed to increase the number of available synapses, leaving still 220 rows and in total 112640 synapse.

The event from the on-chip network are received and decoded by the synapse driver and then passed into the synapse array. Each driver controls two lines of synapses. For each event with a matching address the synapse will fire with a pulse length of t_{syn}. For a PLL of 100 MHz it is $t_{\text{syn}} = 10\,\text{ns}$. For each synapse row it can be selected in the synapse driver, if the synapses are connected to the excitatory, the inhibitory input, or both.

The strength of synaptic events can be regulated by three parameters. For different reference currents the corresponding V_{gmax} are generated by the floating gates. For each synapse row one of these is chosen and can be scaled by g_{div} in the range from 1 to 30. The final current pulse I_{syn} toward the synaptic input is then generated by the synapse. Each synapse has an individual 4 bit digital weight w acting as multiplier.

This gives the input current to the synaptic input

$$I_{\text{syn}} = V_{\text{gmax}} \cdot g_{\text{scale}} \cdot \frac{w}{g_{\text{div}}}, \tag{2.18}$$

where $g_{\text{scale}} = 4$ for revisions 2 and 4, and $g_{scale} = 0.4$ for revision 4.1 of HICANN. A more technical descriptions and a refined model follows in Section 5.3.

Further, the synapses support STP and STDP. Both allow changes of the synaptic weight based on simple rules. A STP mechanism is implemented in each synapse driver and modulates the length of the synaptic pulses. STDP even allows for updating the digital synaptic weights w for each synapse individually. For this, two eligibility traces – one for correlated and one for anti-correlated spikes – are stored in each synapse. These are periodically evaluated and allow changes in the synaptic weight. These features have been evaluated by Billaudelle and Nonnenmacher [Billaudelle 2014; Nonnenmacher 2015].

2.2.4 Analog Parameter Storage

On HICANN each neuron has 21, or 23 in revision 4, individually configurable voltages and currents which control its behavior. In addition, there are 24 voltages and currents that are shared between neurons and are required for other components. These parameters are stored in an analog parameter storage. Its memory cells are implemented using single-poly floating-gate technology and therefore called floating gate cells [Srowig et al. 2007; Millner 2012; Loock 2006], which provide either voltages from 0 to 1.8 V or currents from 0 to 2.5 nA. The technology of the floating gate cells is similar to FLASH memory [Hasler et al. 1999].

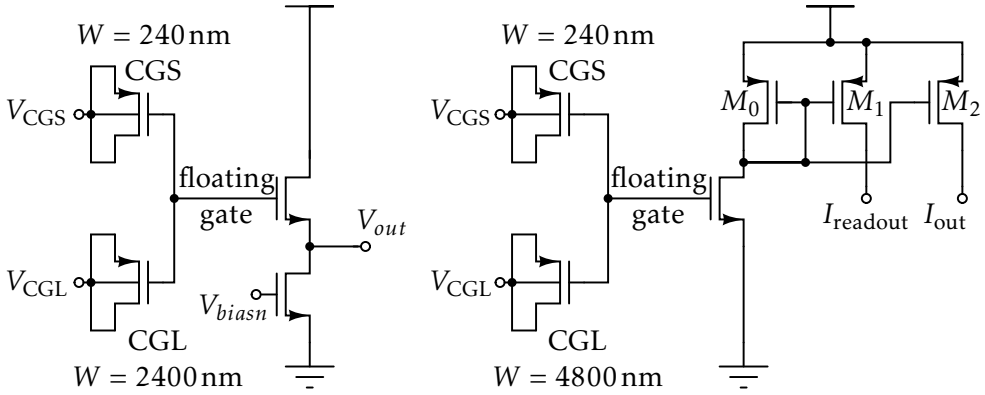


Figure 2.7 Schematic of the floating gate cells as they are implemented in all revisions of HICANN. The charge on the insulated floating gate controls the output V_{out} of the voltage cell and the output I_{out} of the current cell. The floating gate is charged by a tunnelling current, if V_{cgs} = VDD12 on the control gate small (CGS) and $V_{cgl} = 0V$ on the control gate large (CGL). Swapping the potentials causes a decharging current. Figure taken with modifications from [Millner 2012].

Each floating-gate cell has one floating-gate transistor with a fully isolated transistor gate. Its gate can be charged or discharged using Fowler-Nordheim-Tunneling [Lenzlinger et al. 1969]. This requires a sufficiently strong potential, which is provided by the floating gate programming voltage VDD12. To enable sufficient tunneling currents, VDD12 must be at least 10 V. The usual operating range in HICANN is from 10.5 to 11.5 V. Special adaptations were necessary to control VDD12 on the chip in the 180 nm Complementary Metal-Oxide-Semiconductor (CMOS) process used for HICANN. A more detailed description can be found in previous publications [Srowig et al. 2007; Hock 2009].

Figure 2.7 shows the schematic of both cell types. The output of the cell V_{out} or I_{out} is controlled by the charge on the floating gate. Because the floating gate transistor is an NMOS transistor, an uncharged gate has no output and the output increases as electrons are drained from the gate.

To control the charge on the floating gate and to create the necessary electrical field that allows tunneling, two additional transistors are connected to the floating gate: The control gate small (CGS) and control gate large (CGL), which is ten times larger for voltage cells and twenty times larger for current cells then CGS. CGL couples much stronger to the floating gate and pulls it to the desired potential, while the tunneling current flows through CGL. To increase the charge on the floating gate CGS is connected to VDD12 and CGL to ground. To discharge the floating gate the voltages are switched. The current flowing on or off the gate depends strongly

on VDD12 [Loock 2006]. Without active programming voltages the cells are stable over several hours [Millner 2012].

In voltage cells the final voltage is created by a source follower. It transforms the current generated by the floating gate transistor into the output voltage V_{out}. The output range of the voltage cell can be controlled by the bias voltage V_{biasn}, which can have 16 fixed values.

In current cells, a current mirror, built by M_0 and M_2, provides the output current I_{out} of the cell. The reference current for programming is generated by a second current mirror formed by the transistors M_0 and M_1. It generates a four times larger current. For various currents additional scaling factors are implemented [Millner 2012]. We use only the scaling of control parameter of the leakage OTA I_{gl}, which can be sets to be $1:1, 1:3, 1:27$.

The design allows to create a voltage cell with just 5 transistors and a current cell with six transistors. This makes them sufficiently small to provide the large number of parameters required on HICANN.

The analog parameter storage is organized in four blocks. Each block has 24 rows of 129 floating gate cells, where the even rows provide voltages and the uneven rows provide currents. In total 12 384 parameters can be stored. Each row provides one globally used parameter and 128 neuron parameters, but some rows are unused. The assignment of the parameters to the neurons is documented in [SP9 Spec 2015].

The programming of the floating gates is done via an on-chip controller. Thereby, the cells are programmed row-wise and can either be charged or discharged in one programming pass:

First, a 10 bit target value is loaded into the controller for each cell. Then the controller compares the value of each cell against a reference value generated by a Digital-to-Analog Converter (DAC) based on the target value. If the value is too low (or too high in the case of discharging) the programming voltage is activated for a short period. The controller stops either after each cell is above (or below) its target value or after a maximum number of cycles. It has a set of digital parameters controlling the programming process. The *maxcycle* parameter limits the maximum number of programming cycles per pass, while the *readtime* parameter controls how long the controller waits before the output of the floating gate is compared to the reference value. If the wait is too short, it distorts the comparison because the readout line needs to charge to the correct value. Based on the comparison result the controller decides for each cell if an additional charge or discharge is needed. The duration of the charging pulse is controlled by the *writetime* parameter. The *accelerator step* parameter doubles the *writetime* parameter, after each number of *accelerator step* programming cycles.

The fixed (dis-)charge times can cause an overshooting over (under) the desired value. In order to minimize deviations from the target set values, we use multiple passes of alternating charging and discharging cycles. We evaluated the precision of the floating gates in Section 5.1.

2.2.5 Readouts

The HICANN has two 50 Ω-terminated outputs for analog voltages. In the BrainScaleS System each of the outputs shares the connection with the outputs from 8 HICANNs. This limits the number of voltages that can be read simultaneously to $384 \cdot 2/8 = 96$. Both have an additional amplifier, which will add a systematic error to the returned voltages. This implies, that voltage read from the two analog outputs are not comparable to each other. We therefore use only a single output per HICANN for calibration tasks. Even, if the offset is only estimated to be 4.5 ± 0.8 mV [Millner 2012].

The DC-termination of the HICANN is actually around 42 Ω instead of the nominal 50 Ω [Millner 2012]. We conducted measurements on each of the 16 HICANNs on the two prototype wafer systems and on 8 HICANNs on wafer 20 and found the DC-termination values to be equally distributed in the range of 41.0 to 44.0 Ω. The resistance varies only slightly for HICANNs on a single reticle. The variation between the different reticles can be caused by two effects, the varying resistances of the whole connection between HICANN and AnaRM and probably less strongly by process variations in the resistor on HICANN. A decision cannot been made yet because the number of samples has been too small so far.

The outputs can read voltages from various sources on HICANN. Primarily they are intended to read out the neuron membranes. Each neuron has an individual amplifier for this signal, which also needs calibration [Schmidt 2014]. But it is also possible to read the following analog signals: The neuron membrane of each neuron can be read with either output and it is possible to read from two neurons in parallel. However, not all combinations of neurons can be read simultaneously. Further, the output of any single floating gate cell can be read. For the current cells this is the output signal for the comparator, which is a voltage. Lastly, the fire signal and the voltage of the DLL lock in the synapse driver can be read for one synapse driver on each half of HICANN. For the details of possible combinations of analog readout and targets we refer to [SP9 Spec 2015].

2.3 System Setup

Each BrainScaleS Systems consists of 384 HICANNs equipped with over 100 000 neurons. It includes supporting electronics, which provide power, channels for communication and configuration, and AnaRM to record analog signals [Müller 2014]. An assembled system is shown in Section 2.3.

We here give a short introduction into the wafer-scale integration, followed by a short description of the FCP and AnaRM. Lastly, we present the Demonstrator Setups, a portable version of the BrainScaleS System, that can also be used to test single-chip HICANNs.

The BrainScaleS System was developed during the BrainScaleS project. At the end of the project, in March 2015, the first 8 systems had been deployed, of which 2 were prototype systems. These systems are now part of the Neuromorphic Physical

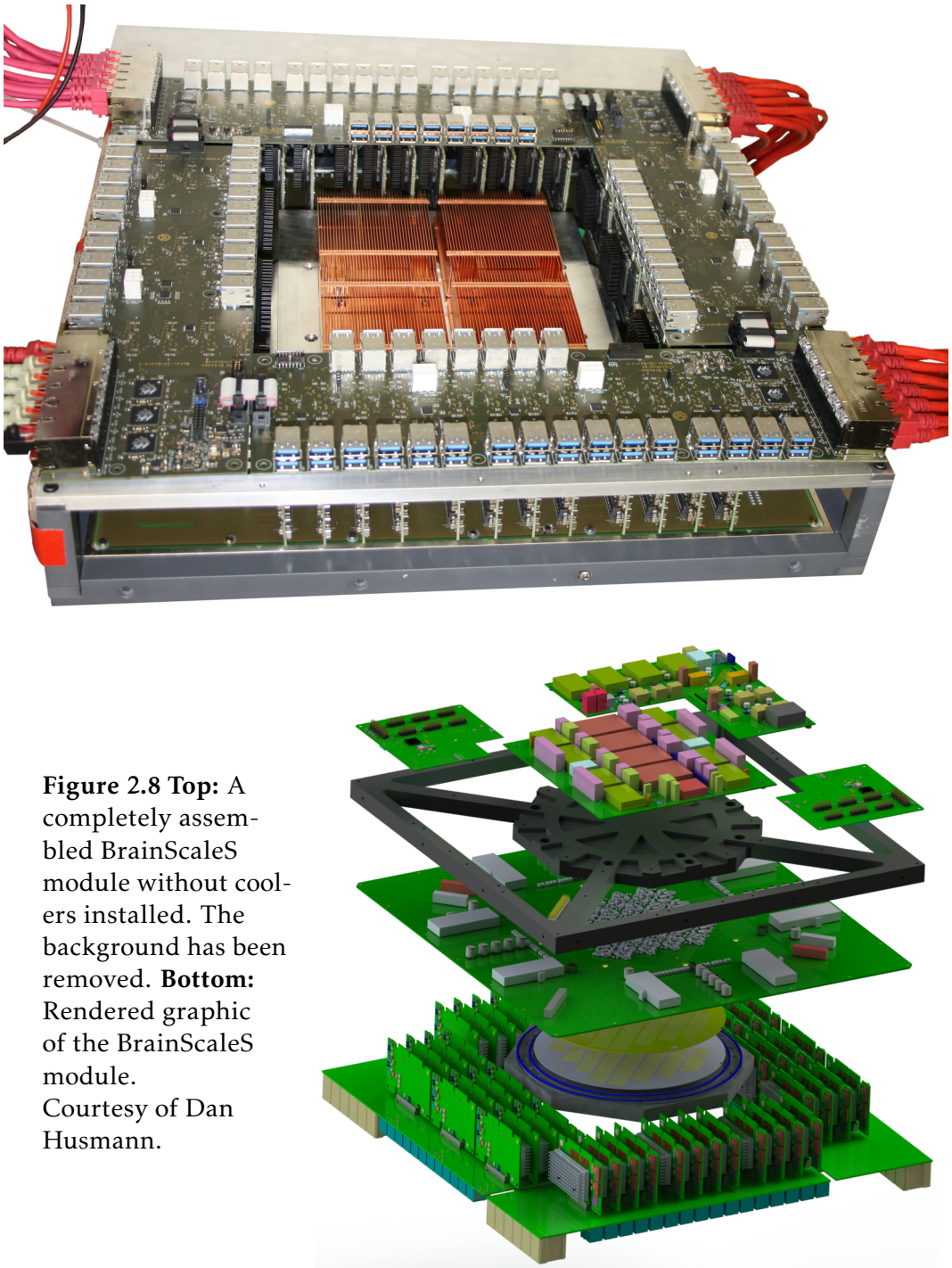


Figure 2.8 Top: A completely assembled BrainScaleS module without coolers installed. The background has been removed. **Bottom:** Rendered graphic of the BrainScaleS module. Courtesy of Dan Husmann.

Model version 1 (NM-PM1) in the current Human Brain Project (HBP). By the end of the ramp-up phase of HBP in March 2016, 20 systems have been deployed.

2.3.1 Wafer-Scale Integration

After production, the wafers are kept intact instead of cutting them into single chips [Schemmel et al. 2010; Jeltsch 2014]. Small areas, called reticle, are separately processed during manufacturing, so not all HICANNs on the wafer are yet interconnected. A post-processing step, that deposes additional metal-layers on top of the wafer, creates the full connectivity as described in Section 2.2.2.

This technique, called wafer-scale integration, allows higher integration densities of HICANNs in the system than regular chip bonding techniques, because the on-chip buses can be more densely routed across chip boarders as with using classical bonds. The horizontal and vertical buses of the on-chip network can now reach over the whole wafer. Furthermore, the wafer-scale integration is more energy efficient compared to externally wired connections.

A possible drawback of this method is that malfunctioning chips - an inevitable site effect of chip production - can not be replaced on the wafer. However, it is possible to route around erroneous HICANNs or to use them partially. Tests over ten wafers have shown that 86.6 % of the HICANNs are digitally functional. But since the rate varies strongly between the systems, further improvements are likely [Mauch 2016].

2.3.2 Configuration and Spike Data

The HICANNs of a BrainScaleS System are controlled via host computers. The communication between them is managed by the pulse communication subgroup (PCS) [Scholze et al. 2011; Thanasoulis et al. 2014]. Compared to the previous publications about the BrainScaleS System, the PCS was updated to use now 48 Xilinx Kintex-7 Field-Programmable Gate Arrays (FPGAs) [Müller 2014]. Of which each FPGA is connected to 8 HICANNs.

The PCS handles the input and output spikes of experiments as well as the configuration of the system. The spikes for an experiment are buffered in the playback-memory. During the experiment it send the spikes with a precise timing to the HICANNs and records the returning spikes. The maximal available rate for sending and recording is 25 MHz per FPGA. Spikes sent by the HICANNs above this rate will be dropped and spikes that are too densely packed can also be shifted.

Currently configuration data is send data directly to the HICANN chips. The transmission is done via the FPGAs, using Host ARQ protocol (HostARQ) and HICANN ARQ protocol (HICANN-ARQ) as transport layer protocols [Müller 2014]. But recently the first implementation of an configuration buffered in the playback-memory was done [Pilz 2016]. In this mode, first the complete configuration is generated and buffered in the FPGA and then written onto HICANN. It actually

allows a faster configuration, because the configuration data can be send with precise timings making many busy waits in the configuration software obsolete.

2.3.3 Analog Readout Module

The BrainScaleS System is equipped with a custom Analog Readout Module (AnaRM). Each AnaRM consists of two modules: the Flyspi board and an expansion board. The Flyspi board is controlled via USB and has an FPGA, which controls communication and measurements. The extension board provides an Analog-to-Digital Converter (ADC) with 8 input channels with 50 Ω DC-termination, 12 bit resolution and a sample rate of up to 125 MHz. To speed up the data transfer only 96 MHz are used.

The analog readout on the BrainScaleS System is limited by the number of readout lines available. 64 HICANNs have to share two AnaRMs, because the analog readouts of 8 HICANNs are merged on the wafer. Special care has to be take in configuration here, because these lines can be driven from all 8 HICANNs simultaneously. Optionally a multiplexer allows replacing 6 AnaRMs with a single AnaRM, so that two AnaRM are sufficient per system.

2.3.4 Demonstrator Setups

Aside from the BrainScaleS System HICANN can be used with the Demonstrator setups [Schemmel et al. 2012; Millner 2012]. These are portable versions of the BrainScaleS System designed to hold up to 8 HICANNs, of which are not interconnected. But it would be possible to interconnect each two of the HICANNs. Aside for demonstration are also be used to test single chip HICANNs and the FPGA firmware.

The first Demonstrator setup, as described by Schemmel et al. [2012] and Millner [2012], reassembled the BrainScaleS prototype. It uses the Virtex-5 FPGA for communication, which was used in the BrainScaleS prototype systems [Müller 2014]. The data for the previous publications about calibration was taken on those systems [Schmidt 2014; Schwartz 2013]. Due to the alignment of the boards, these are also referred to as Vertical setups.

These old Demonstrator Setups were superseded by newer versions using four Kintex-7 FPGAs, like they are used in the BrainScaleS System. These are internally referred to as Cube Setups. Only on two of the four FPGAs HICANNs can be connected, so the Cube Setups can be used with up to 16 HICANNs. Together with Mitja Kleider we installed three of these setups in 2015 in our laboratory to test the new HICANN revision 4 chips. However, for the results we present later on, we repeated these experiments on the actual BrainScales System. These results were not fundamentally different to the earlier experiments.

3 Commissioning of the BrainScaleS System

The BrainScaleS System is a unity of the physical components described in Chapter 2 and software. For the commission of the BrainScaleS System the components need to be assembled together and software needs to be written. While a good part of this has been done, it is a still on going process to improve the system.

The main purpose of the system is emulating neural network models. These are, like in the predecessor Spikey [Brüderle et al. 2011], described in PyNN, which is a common interface for neuronal network simulators [Davison et al. 2009]. This is also used by other neuromophic platforms, like SpiNNaker [Furber et al. 2014; Furber 2016]. We give in Section 3.1 a short overview how the model description is processed and expanded onto the hardware. Hereby, we will focus mainly on aspects towards a calibration framework, as the experimental perspective is well covered by privious work [Jeltsch 2014]. We show, that the operating software allows full and comfortable access to the BrainScaleS System in varying levels of abstractions and we present the Python bindings, which are the base for the calibration framework.

Aside from the software architecture we integrated the AnaRMs into the system. We present in Section 3.2 short characterization of the module and a calibration tool. We show, that AnaRM is a well suited to measure signals from the BrainScaleS System. Lastly the BrainScaleS System has – like any complex System – many possible points of failure. While some have obvious effect others have only subtle ones. These can be found by integration tests of the system, of which we present some examples in Section 3.3. We then close the chapter with a short summary in Section 3.3.4

3.1 Operating Software

The operating software must give control over every aspect of the BrainScaleS System. This holds for the emulation of neuronal networks as well as for more direct access for system tests or the calibration framework. Therefore, it is built up in several layers, each adding additional levels of automation and abstraction of the HICANN configuration. These core components are written in C++ and for many of them Python bindings are created. Python bindings bridge the barrier between the two programming languages and allow the direct usage of the C++ libraries from Python. Potentially destructive settings, like the control of the power supplies, are excluded from this and handled in a separate control software.

Detailed descriptions on the software were already given by Jeltsch [2014] and Müller [2014] and can be found in the specification [SP9 Spec 2015], so we cannot avoid some overlap. We give here first a short general overview of the complete

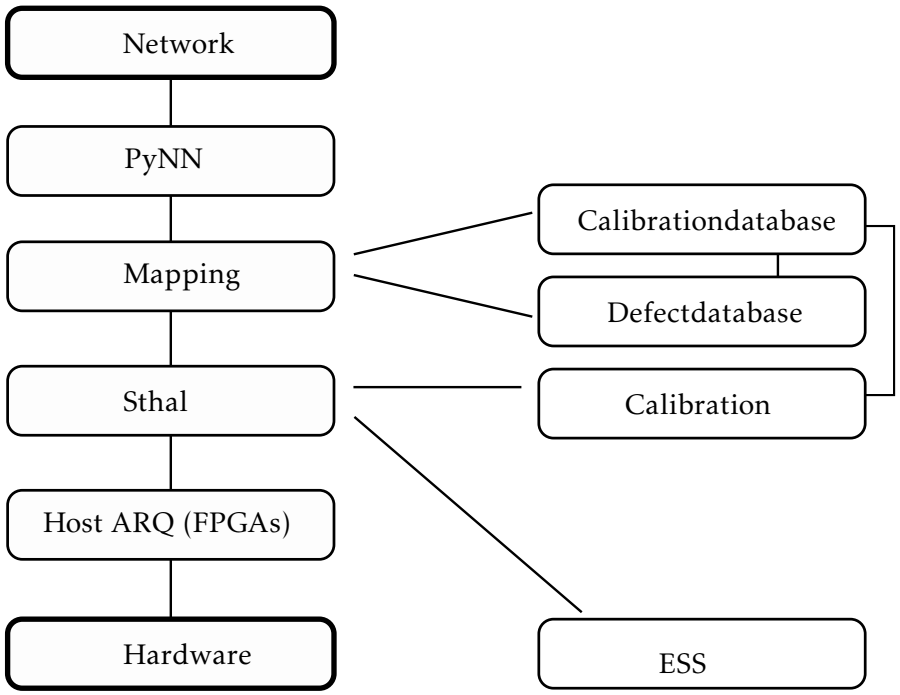


Figure 3.1 Interaction between the various software layers.

operating software. We then focus on the parts relevant for our work: the two hardware abstraction layers, their Python bindings and the calibration framework. We then list our contributions to the software and close with a short summary.

We provide a list of the third party software packages mentioned in this thesis in Appendix E.

3.1.1 Overview

We begin the overview with the main purpose of the system: emulation of neural networks, while the interaction of the components of operating software is shown in Figure 3.1. The BrainScaleS System uses a PyNN compliant interface, which generates a hierarchical model description of a neural network. The mapping software works on this description and transforms the model into a topological equivalent representation of the network for the BrainScaleS System, which takes the physical constrains of the system into account. This step includes also the transformation of the model parameters into the physical parameters required by the emulating circuits.

The results of the mapping are passed to the two hardware abstraction layers. The Stateful Hardware Abstraction Layer (StHAL) provides a complete representation of the available configuration space of the BrainScaleS System. From here the connection to the hardware systems are managed and the actual experiment

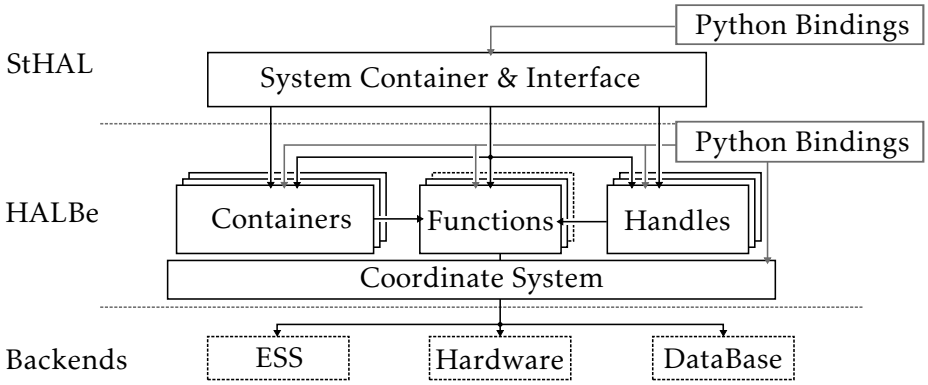


Figure 3.2 Overview of the Hardware Abstraction Layers. The HALbe layer allows fine granular access to the hardware and two different simulation back ends. The StHAL layer combines all the features of the HALbe layer and provides a higher level abstraction. Python bindings are provided for all components. Figure taken from Jeltsch [2014] (modified).

execution controlled. For this the abstract description needs to be converted into the actual bit-configuration of the hardware registers, which is done by the Hardware Abstraction Layer Backend (HALbe). The bit-configuration then are passed to the HostARQ layer and send to the HICANNs.

There are additional helping tools, like the calibration framework, calibration storage and defect management. Further tools for the power management and monitoring of the system are required. Some of these are written in pure Python.

In this work we used the BrainScaleS System either by using Stateful Hardware Abstraction Layer (StHAL) directly or via the calibration framework. For both we often use the python bindings, especially to create simple tools or to do measurements. We now will describe these in further detail.

3.1.2 The Hardware Abstraction Layers

The hardware abstraction layers provide homogeneous access to the hardware system and encapsulate the communication layer. They allow a full control over the system configuration, while increasing the comfort working with the system and adding safeties to avoid common errors. We distinguish the two layers StHAL and Hardware Abstraction Layer Backend (HALbe). The hardware abstraction also allows to transparently substitute the hardware with other back ends. These are the Executable System Specification (ESS) for the BrainScaleS System [Vogginger 2010; Brüderle et al. 2011] and a database back end to record configuration calls. The structure of the Hardware Abstraction Layers is shown in Figure 3.2.

Stateless HAL

The concept of the lower layer HALbe is to provide access to every single component of the chip in the smallest possible granularity, that allows a safe access. Its interfaces can control AnaRM, HICANN and the FPGA. This is achieved by creating an abstract coordinate system and separate container data structures for each component of the chips. We want to illustrate this for a single synapse driver. The possible configuration of the driver is stored in the structure `SynapseDriver` and the specific driver is addressed using the coordinate type `SynapseDriverOnHICANN`. The used back end is represented by the handle type `Handle:HICANNHw`, which represents in this case access to the hardware. Reading and writing the synapse driver configuration HICANN is done by using back end functions:

```
void set_synapse_driver(
        Handle::HICANN & h,
        SynapseDriverOnHICANN const& s,
        SynapseDriver const& drv_row);
SynapseDriver get_synapse_driver(
        Handle::HICANN & h,
        SynapseDriverOnHICANN const& s);
```

This interface approach makes use of the feature, that C++ is a strictly-typed programming language. We do not pass builtin types as function arguments, but encapsulate every information into a separate type. This way we avoid, that arguments can be accidentally switched by the user. Further, this concept allows to enforces range checks directly in the data types. This ensures for example that a coordinate can only point to valid components on the chip. Also the container types implement checks on the possible configuration states and come with sensible default settings for each component. This way back end functions can focus on transforming data to the bit-pattern required for the hardware.

With this approach the attempt to create an invalid configuration directly causes an error and not later when the bit-configuration has been generated and is written to the system. Further details of the implementation can be found in Jeltsch [2014].

Handles

Each back end is represented by different handle types. We provide base types for FPGA, HICANN and AnaRM, which must be inherited by the back end handle types. The handle manages resources used by HALbe. In case of hardware this is the connection to the system. This is created as soon as the handle is constructed and will be closed when the handle is destructed, this is a common C++ ideom know as resource allocation is initialization [Sutter et al. 2004]. If the connection cannot be created the handle creation will also fail. Therefore, the HICANN handle cannot be used independently from the FPGA handle because the communication to the HICANNs is managed by the FPGAs. All required information must be explicitly

passed to the handle on its construction. For example the IP-address is required to request an FPGA hardware handle.

Each possible back end of HALbe is represented by an own handle type, which is inherited from the base type of one of the three base types. The configuration calls are then dispatched by the back end functions to the correct target. We supervised the Bachelorthesis of Pape [2013], who implemented the ESS back end for HALbe. For this back end an ESS instance is created and passed to the handle on construction. It proved to be a valueable tool to test the results of the mapping independtly from the BrainScaleS System [Jeltsch 2014].

Stateful Hardware Abstraction Layer

StHAL organizes the structures given by HALbe hierarchically and can hold – as the name indicates – the complete configuration state of a BrainScaleS System. Based on this data StHAL can provide routines for more complex configuration tasks. An example is to set the analog output for a neuron and ensuring that it is only set for this neuron. This simplifies complex tasks for users and upper software layers significantly.

Additionally, StHAL manages the used hardware resources. An integrated hardware database keeps track of the available BrainScaleS System,Internet Protocol (IP) addresses and the availabilty of AnaRMs for each HICANN. The user can automatically connect and disconnect to an entire BrainScaleS System by a single function call and only the actually used FPGAs and HICANNs are allocated. Further StHAL provides an interface to the AnaRMs. It loads and applies the calibration data for the read voltages and manages the trigger of the AnaRMs during the experiment.

The execution of an experiment managed by StHAL has three steps. First, the desired configuration is created, then the configuration is written onto the hardware and last the spikes are transferred to the FPGA and the experiment is executed. The correct order of configuration is hereby guaranteed. But it is also possible to customize the configuration process to update only a subset of the configuration. Also we can easily rerun experiments using the same configuration with different spikes, as the experiment execution is independent from configuration.

3.1.3 Python Bindings

Python Bindings are already successfully used for Spikey, the predecessor of the BrainScaleS System [Müller 2014; Brüderle et al. 2011]. While for Spikey these were still written manually, we now use fully automated binding generation. It was originally developed for the hardware abstraction layers StHAL and HALbe, but the tools we developed allowe to create bindings for further components, for example for the calibration data, the defect management and the PyNN implementaion [Jeltsch 2014; Klähn 2013].

Python is a versatile script language, which became increasingly popular over the last years in the scientific community. The entry level for data evaluation and

visualization is much lower in Python than in C++. Packages like NumPY, SciPy and pandas allow the efficient handling of numeric tasks and large amounts of data [Pérez et al. 2007; Jones et al. 2001–; McKinney 2013]. Further tools like the Jupyter Notebook and matplotlib can be used to efficiently work on remote machines and to visualize data [Jupyter 2015–; Hunter 2007]. All this allows an easier access to the system for beginners and improves the usability of the system. Further the python bindings are used for rapid development of simple tools and tests, as for example shown in Sections 3.1.4 and 3.3.

For successful usage of Python it is essential, that the bindings cover the complete software functionality. Therefore, we went with an automated wrapping process using the Py++ software package. It can parse the C++ sources and generate code that uses the Boost.Python library, which is then compiled to the python module. We integrated the process into the build chain of the software, so that the bindings are generated after each software change. As Py++ is currently not maintained, we had made some adaption to improve the compatibility to the C++11 standard.

We took special care to keep a common semantic between the C++ interface. This requires considerations on both software sides. For example we need to add hashes and C++ like comparison operators to the wrapped classes, so that they can be used as expected in python containers. We also need to work around the fundamental difference of the assignment in Python and C++. We do this by exposing members using the array operator. By taking some simple considerations into account, it is often possible to write code, that can almost be used identically in C++ and Python. For example, we can set the weight of a single synapse in a StHAL container in both with the following line:

```
h.synapses[SynapseOnHICANN(Enum(324))].weight = SynapseWeight(10);
```

Further we provide special conversion functions to maximize the performance of crossing the language barrier. This is important for large amounts of numeric data and we support NumPY here in both directions. We can pass NumPY directly into C++ using the PyUblas library. In the other direction we created an efficient automatic conversion of C++ standard vectors into NumPY arrays, without the need to copy the data.

Also the PyNN implementation for the BrainScaleS is based on efficient reuse of the Python bindings. For these we implement the PyNN interfaces directly in C++ and generate bindings for Python. We map the reference semantic of Python here by using smart pointers in the interfaces. The complete network structure is efficiently stored in C++ data structures.

3.1.4 The Calibration Framework

The calibration has its own special requirements on the hardware control, that are fundamentally different from the mapping. It works only on single HICANNs and only a very rudimentary routing of spike events is required. On the other hand a very fine granular control over the neuron parameters is required, so that for

example the calibrations of the previous calibrations steps can already be applied. Further the calibration task needs to repeat measurements, sweep parameters and analyze the results. Therefore, a calibration framework was created specifically for those tasks. It is written in Python because it also requires many numerical tasks, for example fitting the parameters of a transformation function to the recorded data. It is build on top of StHAL.

The calibration frame work defines an experiment as a series of single measurements. Usually the experiments do the calibration of a neuron parameter, but they are also used otherwise, for example to characterize the synaptic weights, as for example done in Section 5.3. The parameters for each step are defined by a configuration file. These are mainly the neuron parameters, but also other parameters or spike inputs can be set individually. These measurements are then executed sequentially. For each step the membrane of all neurons is read or the spike output of all neurons is recorded, as required by the current experiment. This is directly analyzed in parallel to the ongoing measurements.

After all measurements steps are completed, the results are evaluated and the calibration is created and stored in the calibration back end. Neurons where this failed are marked as defect.

3.1.5 Contributors and the Author's Contributions

The software described in this section is the work of many minds, which contributed to the BrainScaleS System with code, ideas and tests. We list all contributors in the Acknowledgments.

The authors contribution to the BrainScaleS System were mostly in the software noted above: HALbe, StHAL, the generation of the Python bindings, the PyNN implementation, the calibration framework. Additionally we worked on the build tools and the integration automated unit tests .

3.1.6 Summary

The clear hierarchical structure of the software simplifies testing and debugging. Access to the hardware is possible from every layer in different levels of abstraction and granularity.

The hardware abstraction layers presented in this section provide an easy usable interface for low level experiments and tests on the BrainScaleS System. The type-rich interfaces and consequent range checks allows to catch errors early. Mix-up in addressing components are nearly impossible. In the same time it provides the base for mapping, calibration and tests.

The various back ends allow a versatile usage of the upper configuration layers. Experiments can not only run on the hardware but also in ESS simulations. This enables off-line tests of the complete software stack.

The created Python bindings allow for an easy and intuitive access to the system. It is even possible to work interactively [Tran 2013]. They also allow easy usage of

highly productive tools for numeric task and visualization. Based on these features we created an versatile and efficient tool for the calibration of neurons.

3.2 Integrating the Analog Readout Module

The Analog Readout Module (AnaRM) is specifically designed to record analog signals from the BrainScaleS System. The integration of the boards into the system has been a divided task. The software integration of the board were mainly done by Gorel [2013], while we created – in collaboration with Mitja Kleider – a semi automatic calibration and testing tool for the modules and installed the first AnaRM in the Demonstrator setups. This is necessary because currently about 70 AnaRMs are used. The AnaRM is independently calibrated from the BrainScales System, which allows to easily interchange the boards.

As the voltage source for the calibration, we use a Keithley 2635B sourcemeter connected to AnaRM over a regular 50 Ω resistor to emulate the HICANN 50 Ω DC-termination. We hereby use for all 8 channels the same potential, so we do not have to switch connectors for the calibration of each channel. For the typical calibration run we use 20 steps from 0 to 1.8 V. The whole measurement process is then fully automated[1]. On the returned data a polynomial of 2nd order is fitted and the resulting calibration curves are manually controlled because some boards had defect channels caused by soldering problems. If the result is accepted, the calibration and a protocol of the measurement is saved and can then be used by the regular software. The total process takes less than 5 min and includes the programming of the firmware.

We found that all boards show a homogeneous behavior. We tested the calibration of 8 AnaRMs on all channels. It showed that for an input range from 0 to 1.8 V over a 50 Ω resistor between 2790 and 2807 of 4096 digital values are used. This is about 68 % of the available range and leads to a precision per bit of about 0.6 mV. Figure 3.3 shows the resulting fit parameters and the residuals. All fits have very similar parameters with an offset in the range of 0 to 70 mV and a slope from 2010 to 2050 mV. The residuals of the linear fit are identical for all measured channels. The worst residual value is about −10 mV, which is less than 1 % of the input range. We correct this for the calibration by using a polynomial of 2nd order. The Bachelorthesis of Epp investigated the signal quality of AnaRM in more detail [Epp 2016].

We measured the maximum transfer speed using the HALbe interface of AnaRM with 300 Mbit s^{-1}, as shown in Figure 3.4. This is a bit lower than the theoretical maximum of 480 Mbit s^{-1} provided by USB 2.0, but this depended on the used machine. Of our interest is the communication overhead, that dominates for recordings with less than 1×10^5 samples. We therefore do not use less samples for calibration tasks.

[1] available in the adc-calib repository

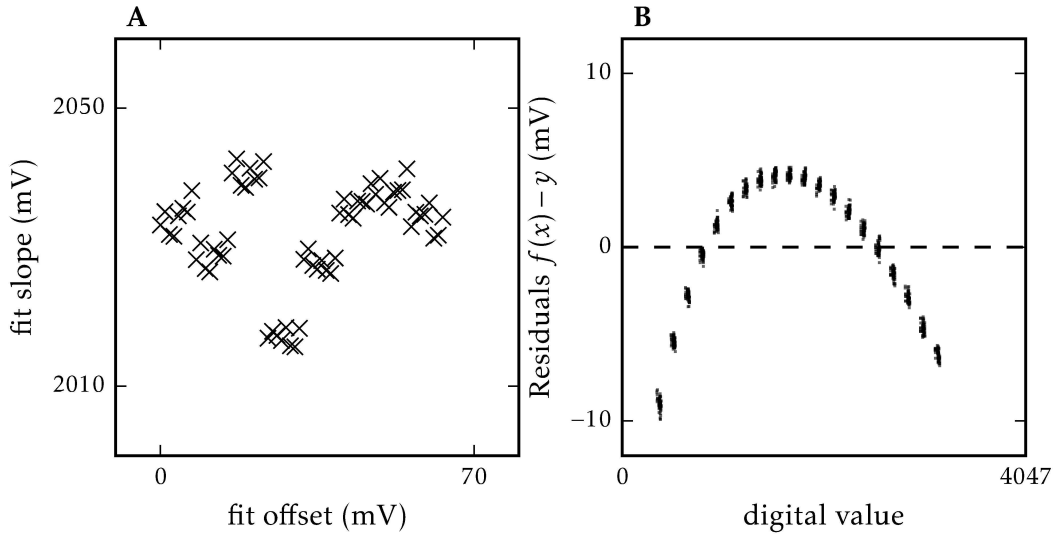


Figure 3.3 A Parameters of linear fits for a sweep on 20 steps with input voltages from 0 to 1.8 V. The parameters were taken from fits on the respond of all 8 channels for 8 different AnaRMs. The groups correspond to the different boards, the variations of the channels on a single board is minimal. **B** The Residuals of all fits shown in (a). The response of the AnaRMs is not perfectly linear, but the behaviour of all channels is uniform.

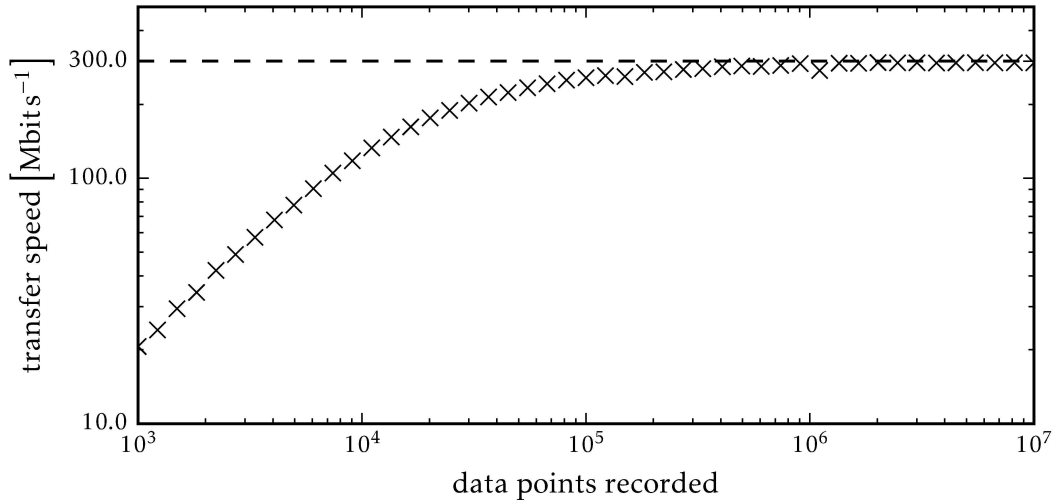


Figure 3.4 Measurement of the maximal transfer rates of AnaRM on a NMPM1 [Müller 2014] host computer using the HALbe Python bindings. For each number of data points the fastest out of 20 measurements is shown. The transfer saturates at about 300 Mbit s^{-1}, for less than 1×10^5 data points the communication overhead dominates the required time.

The method of calibration introduces a systematic error of several percent, depended on the combination of AnaRM and HICANN. These are caused by the offset of the output amplifier and variation of the DC-termination in HICANN, as described in Section 2.2.5. It could be corrected by an separate high-impedance measurement, which requires an additional high-impedance meter as reference. As there are 768 analog channels on a system, this is not feasible. Therefore, the next generation of AnaRM will have the options to use high-impedance DC-termination to allow measurements of high absolute precision.

For our work the relative relation of the voltages to each other are generally more important. Also the developed calibration methods are not sensitive to systematic errors. We therefore conclude that AnaRM is fast enough as well as precise enough to readout voltages from the HICANN.

3.3 Testing the BrainScaleS System

The commission of the BrainScaleS System involves not only creating software and assembling components, but also the testing of their integration. In the search of specific errors we wrote several tools to isolate the problems. Such tests can usually with only a little more effort directly be extended to fully automatic tests and give the benefit that errors can be detected easier in the future.

Additionally, we have driven forward the creation of unit tests for the software by providing an integration of those into the build flow. Unit tests are now executed directly after the build process and the developer is presented with a summary of all tests after the build is finished. Those tests prove valuable to avoid regressions in the software and make contributions by less experienced contributors more save. Lastly, a continues integration server automatically verifies each change, so that all test still pass. This integration was done by Eric Müller. We will further encourage software tests, but this topic is widely covered in the literature, for example in Passig et al. [2013].

While the importance of regular software tests is without question, we here want to focus on integration tests for the BrainScaleS System. Almost everywhere, where different components and the software interact errors may occur, because the system is quite complex. Potential errors cannot always be foreseen and the technological possibilities for direct tests are limited. In the following we give three examples how we work around those limitations to still obtain reliable test results. These tests are designed to prove the correct functionality of the tested components. Locating the source of the error may require further tests on lower levels.

3.3.1 Testing the Pulse Memory

At first, we present a set of tests for the correct functionality of the playback memory, which is described in Section 2.3.2. In this HICANN, FPGA firmware and software interact. While the transmission of configuration data between FPGA an dHICANN

is secured by the HICANN-ARQ and throughly tested [Müller 2014; Karasenko 2014], the transmission of spikes has no such features. It is tolerated by design, that spikes get distorted by transmission errors, because detecting and correcting these events would reduce the possible transfer rate. In any case errors should be very rare and usually not critical for the function [Thanasoulis et al. 2014].

However, we therefore do not have a mechanism during normal operation to verify that spikes are transmitted a the desired point in time and to the correct HICANN. We therefore rely on additional test to verify its functionality.

For testing we rely firstly on the possibility to use a loop-back mode for spikes in HICANN. The eight bus lanes for external spikes can be connected in pairs, so that incoming spikes are directly send back to the FPGA on the neighboring line. Hereby, the spikes are sent and received by the playback memory. In error cases, we cannot distinguish, whether they were not sent, lost in transition or on HICANN or not recorded. Therefore, we additionally rely on the on-chip background generators and the analog debug output of the first two synapse drivers in the center of the chip to pin the location of possible errors.

Receiving Spikes from the Background Generator

We begin with the first test case. Here we program the background generators to emit regular spikes with a given frequency and record them via the playback memory. The spikes pass hereby only through the merger tree of HICANN and are then directly passed to the FPGA. In this setup we can test, if spikes are received at the expected rate and equidistantly. Figure 3.5 shows that the rate of received spikes matches the rate of the sent spikes until we reach the maximal link capacity. The test would pass in this case.

The spikes of the background generators will not arrive with perfect equal distances. This is caused by an erroneous clock-domain crossing between PLL and the highspeed interface in HICANN that operates at 250 MHz. As the time-stamps are generated in the HICANN clock-domain, they can be distorted at this point. Because the less significant bit toggles more often, this is more likely to change causing the time-stamp to jitter around its designated value. We analysed the effect for different PLL settings and the results are shown in Figure 3.6. These timestamps are mostly only shifted a few ns, but we also observed shifts larger than 30 µs. For a PLL 250 MHz about 3 % of the spikes are shifted, but mostly less than 8 ns and for 100 MHz this are below 100 per million. Based on this result we check, that not more than 1 % of the spikes are shifted, if we use a PLL of 100 MHz.

Therefore, we consider this effect in our test and add an error margin to the tolerated number of spikes that deviate from their designated time-stamp. In the test we use all eight background generators with random addresses and rates up to 200 kHz. Because multiple background generators are used, it is possible that two or more generate a spikes at the same time. In such a case on of the spikes is also shifted backwards. In this setup this is as likely as shifts caused by the clock-domain bug.

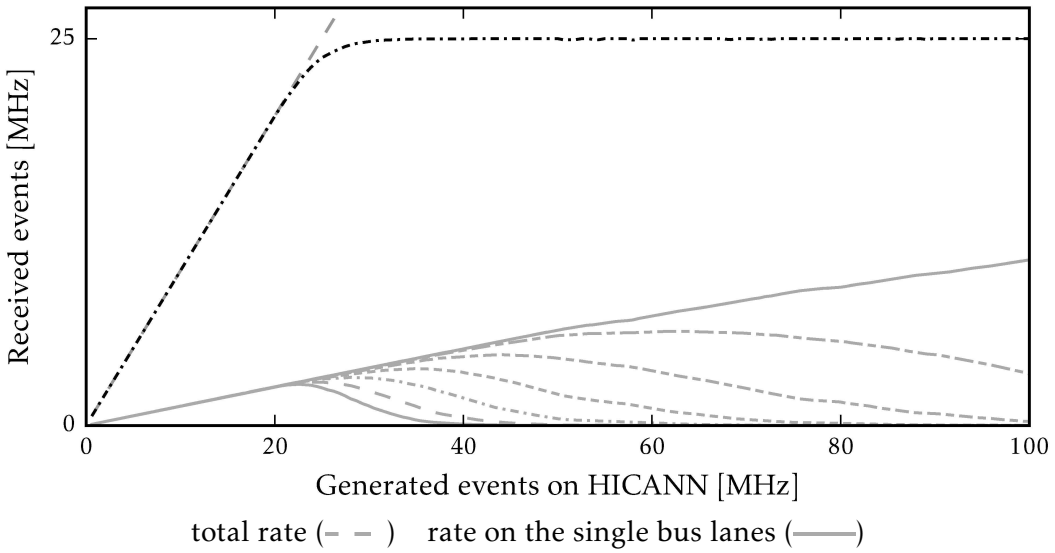


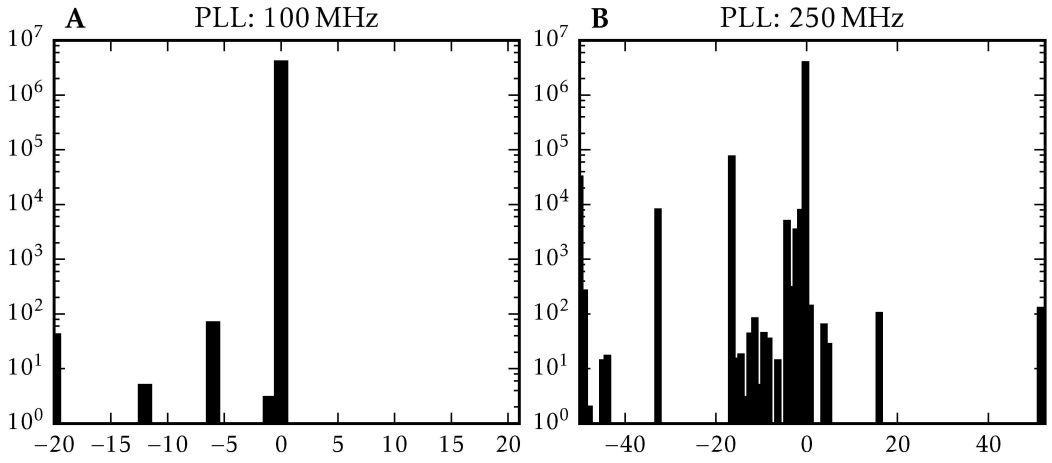
Figure 3.5 Transfer rate of the FPGA for random events generated by all 8 on-chip spike sources. The events are passed from the corresponding 8 output buffers on HICANN to the FPGA. The black line shows the total achieved transfer rate the gray lines the transfer rate of the different output buffers. The link saturates at 25 MHz, but at about 20 MHz spikes can get lost. For larger spike rates single output buffers can be suppressed completely.

Sending Spikes and Compare to Analog Recording

The next test verifies that spikes are actually sent by the playback memory. For two of the synapse drivers the enable signal for the synapses can be monitored using the AnaRM. We can use this to verify the address and time of the sent pulses as well as the functionality of the trigger. Hereby additional spikes from the background generators are required to lock the synapse drivers ahead of the pulses. We limit this test to 20 000 pulses to keep the recording time short.

Looping Spikes

At the end we test the playback memory using the loop-back mode on HICANN. Here we send 1, 2000, 100 000 and about 33 million spikes, which is the maximal capacity of the memory. This covers both extremes in the possible number of spikes. For the lower spike numbers, we repeat the transfer of the spikes and execution of the playback memory 200 times without reconfiguring the HICANN. Lastly we execute an empty playback memory to ensure that no relics remain in the pulse memory from the previous runs.



PLL [MHz]	100	125	150	175	200	225	250
jittered [per million]	28.9	710.1	8.4	0.0	0.0	831.9	29757.9

Figure 3.6 The spikes from the regularly firing can be randomly shifted. This is caused by a not synchronized register, when the spikes are passed from HICANN to DNC. About 4 million spikes were transfered. This plot shows an quite old exemplary measurement on a Demonstrator setup. The results may vary on the BrainScaleS Systems.

3.3.2 Synapse Driver Tests

The next test verifies the synapse drivers of HICANN. We created this in cooperation with Sebastian Schmidt and Sebastian Billaudele [Billaudelle 2014].

The synapse driver decodes the signals of the on-chip spike routing. Spikes are transmitted using a low-voltage differential serial encoding. The lower and upper signal level is given by V_{ol} and V_{oh}. Both are provided by the power supply of the BrainScaleS System and equal for all HICANNs. They need to be set correctly to ensure that spike transmission is reliable. For suboptimal settings a large number of synapse drivers may not work correctly.

Signals can be passed by the on-chip communication arbitrarily on and between HICANNs. Therefore, arbitrary phase shifts in the signal can occur, depending on the actual source of the signal. They are compensated by a Delay-Locked Loop (DLL), that locks on incoming pulses to detect the correct phase and clock-rate. The DLL requires regular spike events at a low rate and with address zero to hold the lock. These are normally provided by the background generators. The lock can be reset to a default control voltage given by the analog parameter storage. The required initial value depends on the used PLL setting.

The test needs to use the neurons to verify the correct address decoding of the synapse driver. We program the synapse driver and the synapse to send spikes to

one neuron per 64 addresses. The neurons are configured to directly fire if they are stimulated by one or more spikes. However, especially on revision 2, this is not possible for all neurons. We solve this by selecting neurons with a good response in advance.

When the neurons are configured, we begin with the actual test. First, the on-chip network is configured to send background spikes and external spikes to a single driver. For this driver the DLL locking is reset. Afterwards we send 64 blocks of 100 spikes with the same address and a distance of 2 ns covering all addresses. The response of the neurons is then matched to the input spike pattern, a typical evaluation result is shown in figure Figure 3.7. Because the neurons often fail to react to spike input, we count only wrongly detected spikes as errors.

Figure 3.8 shows the number of working synapse driver dependent V_{ol} and V_{oh} for both HICANN revision 2 and 4. In revision 2 the optimal working range of the parameters is rather small and it is important to set them correctly. Even then about 10 % of the drivers do not work. This was strongly improved in revision 4 of HICANN, where usually all synapse drivers work for a wide range of V_{ol} and V_{oh} settings.

3.3.3 Connectivity Test for the Analog Readout Module

As a last integration test we present the test of the correct integration of all 12 AnaRMs into the system. The test can detect faulty components, wrong entries in the connection database and a too high noise level. Additionally, it can defect analog outputs of HICANN, but this did not occur yet.

The test works on a single reticle using all eight HICANNs at once and both AnaRMs connected to those. We can test the connections only against a voltage source on HICANN and for detecting a wrong wiring we need to change the output of this voltage source. We therefore use two floating gate cells as sources and connect each to one of the two analog outputs for each HICANN. We have to take care that for each output only one HICANN on a reticle can be active.

The test protocol is the following: At first, all floating gates on all eight HICANNs are written to zero. We then measure for each HICANN voltages of the test cell for both analog outputs. Then we update the voltages on the first HICANN for the cell on the first analog output to 0.50 V and for the second output to 1.0 V. These are then read back and the procedure is repeated for all eight HICANNs. Afterwards we read back the complete HICANN configuration as far as this is possible. Based on this data a report is generated listing all errors that occurred, generating a plot of all recordings and an overview plot visualizing the state of the complete BrainScaleS System. An exemplary overview plot is shown in Figure 3.9.

3.3.4 Summary

System and software tests are indispensable tools for a system of a size like the BrainScaleS System. With the integration of automated test we encouraged and

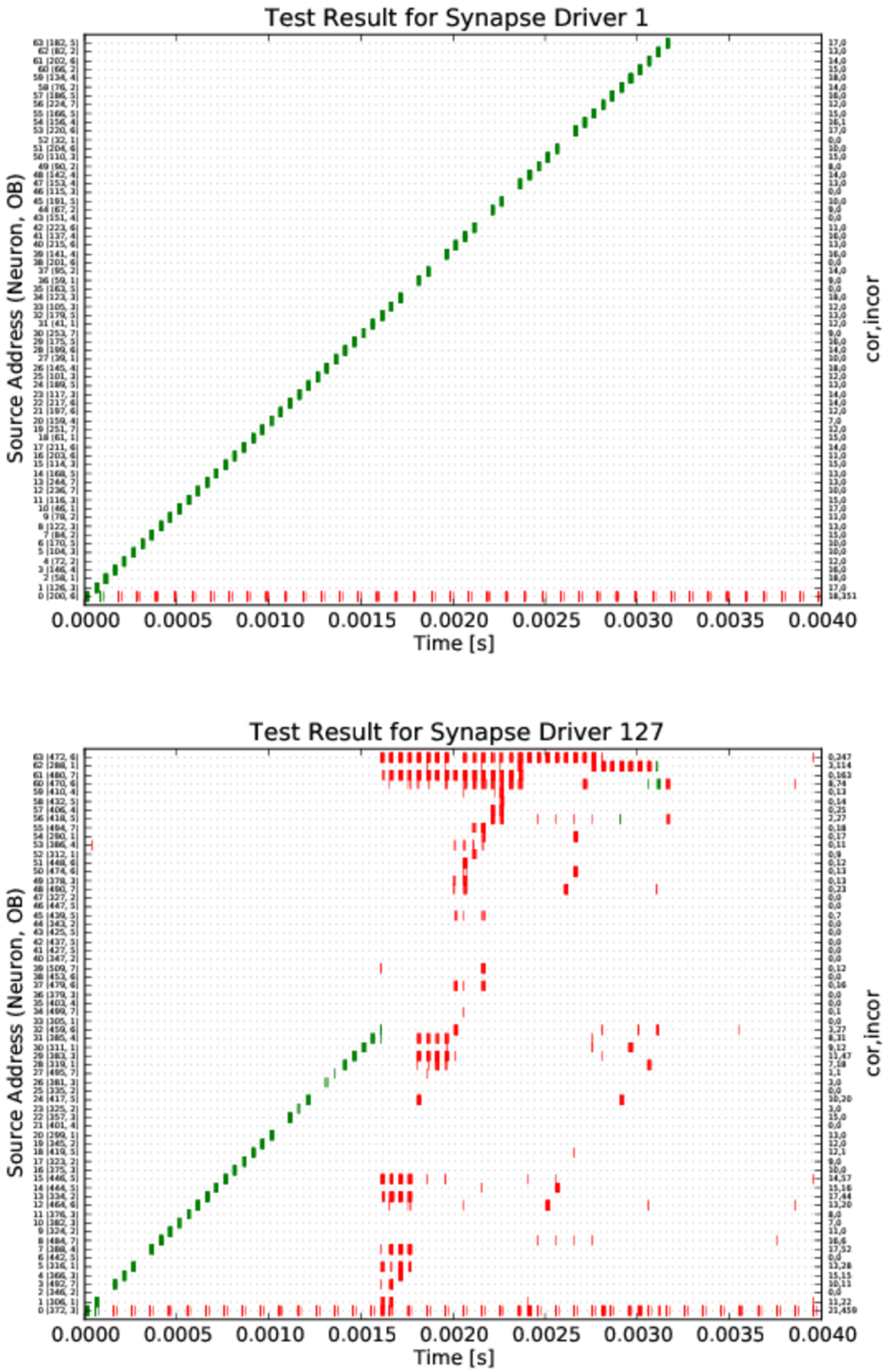
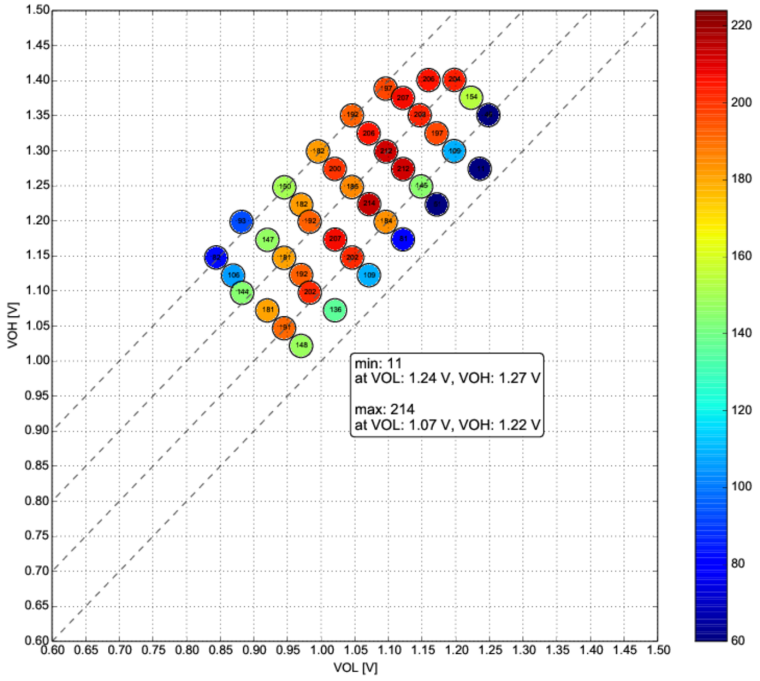


Figure 3.7 Examples for a good and a bad synapse driver. Spikes matching the input pattern are green and not matching are red. In the bottom line the ouput of the background generator can be seen. Courtesy by Sebastian Schmidt [SP9 Spec 2015].

HICANN revision 2



HICANN revision 4

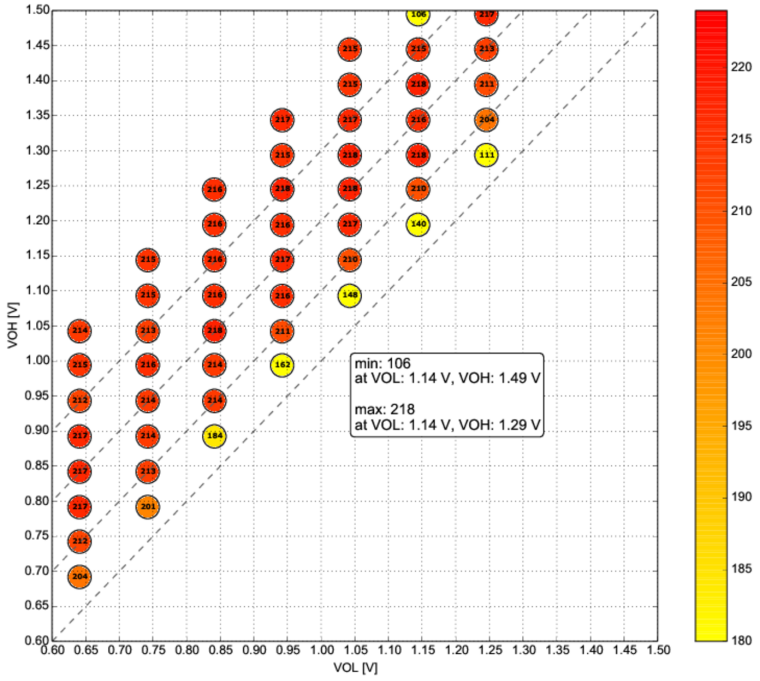


Figure 3.8 Result of the synapse driver test on HICANN revision 2 and 4. The test was conducted on a Demonstrator Setup. Courtesy by Sebastian Schmidt [SP9 Spec 2015].

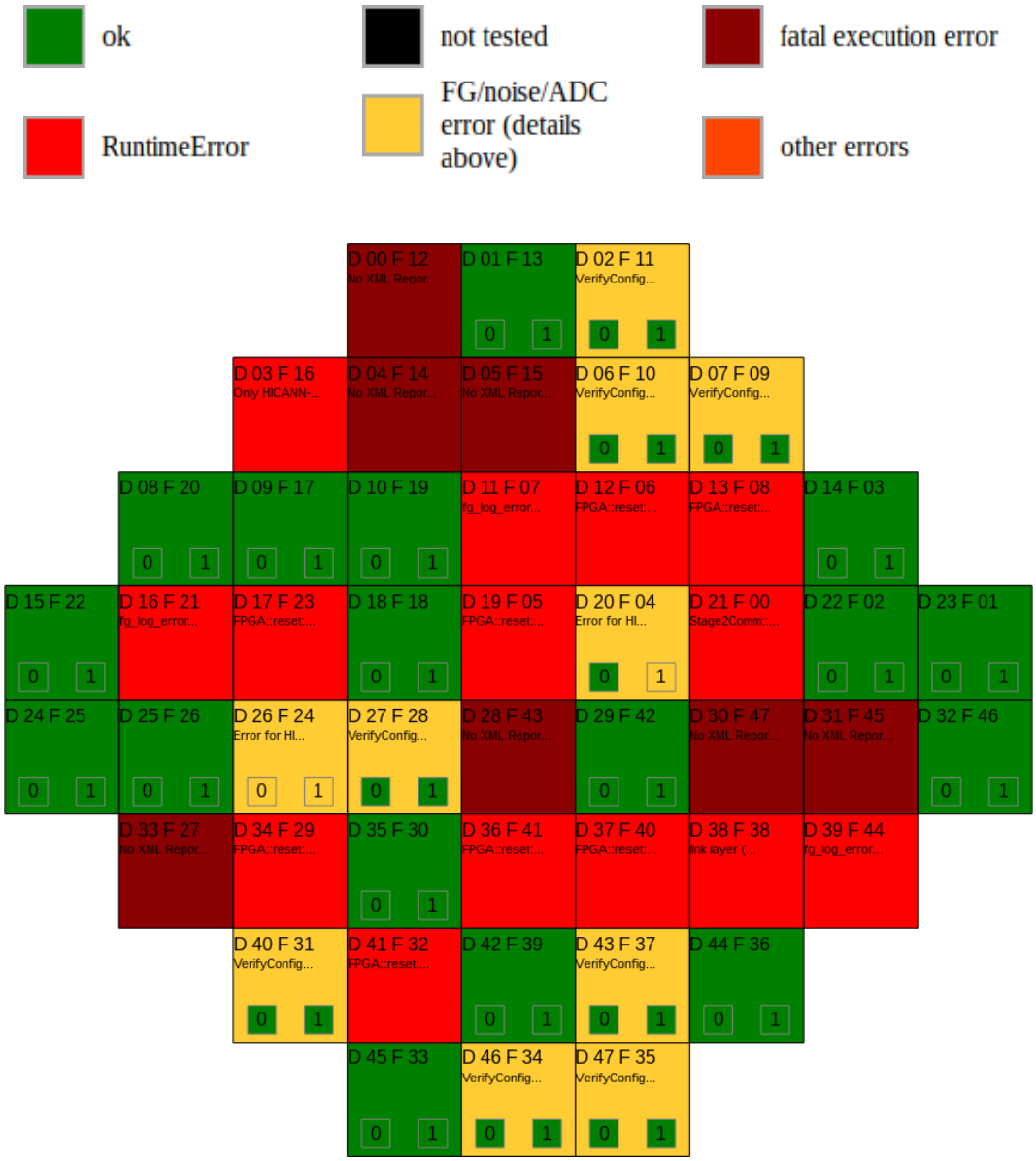


Figure 3.9 Sreenshot of the AnarRM test result visualization. The test shown was exectued on BrainScaleS System W_F01, which is a test system to optimize the assembly of the module. Therefore we can see the various error the test can detect. The status of the analog readouts is indicated by the color of the two smaller squares containing the channel number 0 or 1.

simplified writing software tests. The three presented examples for integration tests are quite complex. But the limited debug feature of HICANN do call for such complexity, which itself increases the risk of introducing errors in the test itself. A more debug friendly design of HICANN would have helped in the commissions of the system, even though such features require area on the chip and development time. This should to be considered for future generations of the system. Non the less are the presented integration tests a valuable tool detecting errors and improving the system over time.

4 Methods

After we presented the BrainScaleS System and its commission we move forward to our main task: the calibration of the synaptic inputs. We begin the chapter in Section 4.1 with a definition of a calibration and the resulting parameter transformations. Afterwards we will follow up with the technical details about the HICANN configuration and measurements in Section 4.2. In Section 4.3, we then present the methods we developed to obtain the time-constants from PSPs. As the measurements are not always sufficient to understand the circuit behavior, we also used simulations of the circuits at transistor level to complement our measurements. The simulation setup is presented the simulation setup in Section 4.4. We conclude with a short summary.

4.1 Calibrating and Converting Parameters

There are two aspects to the calibration of the neuron parameters . Firstly, we need to match the circuit behavior with the desired neuron behavior. Secondly, the devices in each circuit are subject to variations. Especially in analog circuits, this can cause substantial deviations from the desired behaviour [Razavi 2001]

Both aspects have been subject to previous work [Schwartz 2013], but without taking the synaptic inputs into account. We now give a short introduction to the effect of mismatch and related variations and then describe our method for parameter translation.

4.1.1 Device Mismatch and Variability in Integrated Circuits

Mismatch is the time-independent random variation of the physical quantities in identically designed circuits [Pelgrom et al. 1989; Lovett et al. 1998]. It is caused by variations that occur in the fabrications phase of the devices. The properties of many structures, especially transistors, are determined by their size. Here smaller structures are stronger affected by mismatch than larger ones, because the variations get larger relative to their total area. Additionally, various other variations can occur between different production batches, these are summarized as process corner variations [Weste et al. 2011]. Each integrated circuit is affect by these effects.

For transistor devices, these variations show in the current-voltage characteristics. Digital circuits will exhibit different drive strength and thus timings. With analog circuits, one often relies on their precise behaviour. Thus, special measures are required to compensate for mismatch and other variations [Razavi 2001]. For example operational amplifiers or comperators might require a compensation of their input offset voltages or current mirrors of their scaling factor.

In contrast to the mismatch and process corner variations, which are time-independent, the performance of circuits is also dependent on the temperature and the supply voltages. However the BrainScaleS System is designed to provide a constant temperature and stable supply voltages. Therefore we can assume that these are not an issue for the calibration task.

In HICANN, mismatch can be directly observed in the behavior of neuron model. The ranges of the model parameters will vary for each neuron. However, the electrical parameters of the neurons are designed to compensate these effects, so that for each parameter a safe and sensible operating range can be reached.

4.1.2 Parameter Transformations and Ranges

For operate the neuron model in the desired range, we need to map the analog parameters to the parameters of the model: We define a parameter transformation T for HICANN as the mapping of value x for a neuron parameter p onto the digital control bits N_{DAC} of the floating gate controller. Formally we denote it as

$$T_p(x, \vec{a_n}): \quad (p_{\mathrm{min,n}}, p_{\mathrm{max,n}}) \to (0, 1023) \in \mathbb{N}, \tag{4.1}$$

where the domain of the functions $p_{\mathrm{min,n}}$ are $p_{\mathrm{max,n}}$ the possible range of the parameter for neuron n and $\vec{a_n}$ are the parameters of the transformation for neuron n. We require a monotonous relation to avoid ambiguities when applying the transformation. The implementations of transformation functions will have a floating point value as output, which is rounded to obtain an integer value passed to the controller.

As described in Section 2.2.4, the digital controller of the parameter storage tries to set the corresponding output current I_p or voltage V_p based on the digital parameter N_{DAC}. We separate the transformation of the digital parameter ranges of the parameter storage in a ideal linear component and a correction function f_c, which describes the deviation of perfect linearity transformation. We obtain the outputs

$$I_p(n_{\mathrm{DAC}}) = N_{\mathrm{DAC}} \cdot \frac{2.5\,\mu\mathrm{A}}{1023} + f_c(N_{\mathrm{DAC}}) + \epsilon_t; \quad (0, 1023) \to (0, 2.5\,\mu\mathrm{A}) \text{ and} \tag{4.2}$$

$$V_p(n_{\mathrm{DAC}}) = N_{\mathrm{DAC}} \cdot \frac{1.8\,\mathrm{V}}{1023} + f_c(N_{\mathrm{DAC}}) + \epsilon_t; \quad (0, 1023) \to (0, 1.8\,\mathrm{V}). \tag{4.3}$$

Hereby is ϵ_t the error of an programming trial, which is due to the design of the controller of significance [Kononov 2011].

In addition the systematic deviation from a perfectly linear transformation, the design of the floating gate cells, shown in Figure 2.7, adds a per parameter variation to the output value of cell. Especially the current mirror in the current cells is susceptible for mismatch as well as the scaling factors for the various current parameters [Millner 2012, Fig. 3.35]. Therefore we include f_c implicit into the transformation function $T_p(x, \vec{a_n})$ for the given parameter.

Lastly, we take up again the transformation from biologically parameterized

model onto the hardware, which we already brought up in Section 2.1.3. The conversion is a simple linear transformation, dependent on the desired speed up factor of the emulation and the voltage range, that shall be used [Schwartz 2013; Schmidt 2014] For two reasons, we consider this as additional step and do not include it into the transformation $T_p(x, \vec{a_n})$. Firstly $T_p(x, \vec{a_n})$ shall remain a direct relation between the parameter and the in hardware measured quantity and secondly the conversion from biologically conversion has several degrees of freedom, which we do not want to restrain prematurely.

4.1.3 Calibrating a Parameter

A transformation function T_p for a parameter p can be obtained using different methods. The simplest method is to scan the whole range of the parameter domain and do step-wise approximations. This requires multiple measurements per step to achieve a suitable calibration, because due to the trial-to-trial variations the measured value need to be averaged over several trials.

As we want to reduce the number of measurements, we first use T_p as a parameterized model function, that can be fit to the measured data. We then measure for different settings $\vec{y_p}$ of the input parameter p and obtain the corresponding model parameters $\vec{x_n}$ for each neuron. For those, we calculate the transformation parameters $\vec{a_n}$ that minimizing the residuals

$$\|t_p(\vec{x_n}, \vec{a_n}) - \vec{y_p}\| \tag{4.4}$$

for the neuron n using the Levenberg–Marquardt algorithm as provided by LMFIT.

Each measurement is hereby inflicted by the trial-to-trial variations of the chosen, which are propagated to parameters $\vec{a_n}$. This can be improved by either reducing the trial-to-trial variation or by repeating single measurement steps. In the case of normally distributed errors and linear functions, its equivalent to increase the number of measured points. For non-linear functions we still need to make sure that the whole parameter range is well covered by the measurement. If the distribution of the measured values is no longer normal, but skewed, systematic deviations can occur. However, using a parameterized model function allows us to reduce the number of required measurements significantly compared to a step-wise approximation.

The total error of an validation measurement is given by three components: the error of the measurement itself σ_m, the error caused by trial-to-trial variabilities σ_tt and the error of the transformation function σ_tr. Hereby, σ_tt is not only caused by the variation of a single parameter, but also through the interaction of different parameters. As we actually cannot distinguish between the first two, we combine them to the trial error

$$\sigma_\mathrm{t}^2 = \sigma_\mathrm{tt}^2 + \sigma_\mathrm{m}^2 \tag{4.5}$$

under the assumption that both are normally distributed. However, we can minimize

σ_t by repeating the measurements for a given parameter multiple times.

The trial error sets an upper limit to the required precision of the transformation function of

$$\sigma_{tr} < \sigma_t. \tag{4.6}$$

This concludes the definition of the calibration task. We now follow up with the technical implementation of the measurements on HICANN.

4.2 Measurements on HICANN

The HICANN measurements shown in this work were mostly taken using the calibration framework described in Section 3.1.4. It implements a convenient interface to sweep parameters and full low-level access to the hardware based on StHAL. The only exception is the characterization of the floating gate parameter storage, which is done directly using StHAL and shown in Section 5.1. Our work leads to continuous improvements of the configuration process and we take care that these are implemented into the used software for general usage by other. Therefore all the methods presented are available StHAL or the calibration framework. The BrainScaleS Systems used in this work are listed in Appendix B.

We first give a short introduction in the configuration order used for experiments. This is followed by a description of the default configuration used for our measurements. Afterwards, we specify the measurement protocols.

Here the same reasoning applies, as for the development of the software as discussed in Section 3.1.5.

4.2.1 Order of Configuration Routines

All subsystems of HICANN have to be configured in a consistent order to allow reliable and repeatable experiments. We here present the canonical configuration order we developed for StHAL and the reasoning behind it.

The uppermost goal is that two configuration run lead to an identical configuration as far as this is possible given the variations of analog parameters. We therefore configure the system in the beginning of a measurement form the ground up. This is not feasible in the given that usage by multiple user can interleave and the configuration might also decay. However, inside a single experiment differential updates are safe and can easily be implemented.

The following configuration is done in a single pass using the unbuffered communication mode via HostARQ and HICANN-ARQ, however some steps currently still use the Joint Test Action Group (JTAG) interface. Very recently the playback memory buffered configuration became available [Pilz 2016], which is not yet used in the calibration framework. Further we worked only on single HICANNs, so no parallel configuration was used either. The details of the configuration steps are

implemented in the corresponding HALbe function, while the order of calling them is part of StHAL.

Before we start the HICANN configuration itself, the used FPGAs and HICANNs are reset, the PLL of HICANN and high-speed links are initialized [Scholze et al. 2012]. Afterwards the initialization of the HICANN is done, where the custom Static Random Access Memorys (SRAMs) are set to zero. The correct details of this procedure are rather complex and are documented in HALbe. After these steps the configuration can be written, which is done in the following order:

1. Write floating gates and current stimuli.
2. Write synapse configuration.
3. Write the neuron configuration bits, but disable the spike output to avoid spiking neurons from interfering with the configuration process.
4. Configure on-chip communication including merger and background generators.
5. Issue a read command to each tag to ensure that all packages are send (`HICANN::flush`).
6. Lock repeaters and configure synapse drivers (includes locking).
7. Write the actual neuron configuration bits.
8. Configure analog readout.
9. Issue again a read command to each tag to ensure that all packages are send (`HICANN::flush`).

This order ensures that all required analog parameters are available to other components. This most relevant for the locking of the synapse drivers and repeater [Ziegler 2013]. When the current stimulus is enabled, we ensure that the *bias* and *biasn* settings of the floating gate controller are preserved, in the case that reset and initialization were omitted. Because the neurons can be set to spike permanently with very high rate, we disable the spike output of the neurons for the rest of the configuration phase to avoid interference. Afterwards we prepare the locking process for repeaters and synapse drivers. These require a constant spike input to adapt to the signals [Schemmel et al. 2008], which is provided by the background generators. Therefore first the on-chip communication is set up and afterwards we trigger the locking process.

Lastly we have to ensure, that the configuration process is actually completed. This can be delayed, because the transport layer protocolls HostARQ and HICANN-ARQ can buffer data [Müller 2014; Karasenko 2014]. However, a receiving the answer to a read command sent to the HICANN ensures, that all previous configuration commands were processed by HICANN. This is especially important for incremental changes in the configuration in combination with analog measurements, because otherwise the wrong target may be measured.

4.2.2 System Initialization

We conducted all measurements on a BrainScaleS System, where all HICANNs were powered. After powering the systems the control voltages are changed to the values denoted in Appendix A.1. Afterwards all HICANNs in the system are fully configured to the default settings given by StHAL with all analog parameters set to zero. For this we use the `sthal_init_reticles.py` tool. If the high-speed link initialization fails, it uses a basic initialization using JTAG communication protocol. This is also needed for the two reticles in the center of the wafer, that operate without high-speed connections.

4.2.3 Default Settings

For the measurements shown in the work we used the calibration framework, which we described in Section 3.1.4. We conduct our measurements with the PLL set to 100 MHz. We further only single neurons are used with deactivated inter-neuron connections. All neurons have their firing mechanism enabled, but disabled spike output. The firing mechanism is required because the neuron membrane has a stable fix-point at about 1.2 to 1.3 V caused by the operating rage of leakage OTA [Millner 2012]. Without firing the membrane can get stuck at this point until the firing mechanism is activated again. We use the large membrane capacity. Further, we use for the current parameters the default scaling settings [Millner 2012, Table 3.6], but other setting can be selected in the calibration framework as well. The default analog parameters and also the floating gate controller settings are given in Appendix B. We also list the deviating settings for measurements there.

Spike stimuli are generated using the background generators in regular mode. They have the advantage to provide absolutely precise inter-spike intervals. The on-chip routing of spike events on HICANN requires us to use a separate generator for each half. The neurons on each half of HICANN receive the same spike input in parallel using the same synapse driver, so that all synapses in a row are active at once. The usage of up to four spike generators per half is also possible, but their activity cannot be synchronized using the unbuffered communication mode.

4.2.4 Measurement Protocol

The calibration framework can also be used for general experiments, that require the modification of one or more parameters. For each measurement or calibration firstly the basic configuration is defined and then the configurations fore each step are created, which modify the basic configuration.

Here we distinguish between sequential and incremental measurements: the sequential measurements execute each step completely independent from the previous one. For each step, the connection to the HICANN is renewed and the complete configuration is done as described in Section 4.2.1. This method has the advantage, that after each step the measurement can be resumed, if a communication error occurs. Opposed to that the incremental measurement the basic configuration is

done only once. Afterwards the connection is held for each step and only a differential update of the modified parameters is done. We developed this mode for the calibrations and measurements shown in Chapters 6 and 7.

For each step the membranes of all neurons are recorded sequentially one after an other. We hereby use for all neurons the same analog output, otherwise the different offsets of the output amplifiers would make the results not comparable, as noted in Section 2.2.5. Firstly the digital neuron configuration is updated for each neuron to enable its analog output. Afterwards the membrane is recorded for the specified time. As we only use the background generators no triggering is required. The analysis of the recording is then done in parallel to extract the required features for the given task, while further neurons are recorded. After completion of each step the results are stored on disk.

The complete measurement can then be used to create a calibration for each neuron. The changed parameters and the extracted features are used to fit the transformation function to the data, as described in Section 4.1.3. The resulting transformations are then saved for later usage. For the calibration and characterization of the synaptic input we need a special method to analyze PSP.

4.3 Analyzis of Postsynaptic Potentials

PSPs are the only available source of information from the internals of the synaptic input on HICANN, because we can only read out the membrane of the neuron and not other internal voltages, like the integrator of the synaptic input. Fortunately we can rely on an Equation (2.13) describing the PSP, as model for our analysis. Singular PSPs are to much afflicted by noise to be analyzed directly, so that we need to first average several PSPs and only then can fit the PSP to obtain the synaptic time-constant.

4.3.1 Averaging of Postsynaptic Potential

The base noise level on the current BrainScaleS System is too large to work directly with a single PSP. And aside from the Gaussian-noise on the read traces, also distinct signals from other components of the BrainScaleS System can be observed, which might also affect the analysis. Figure 4.1 shows a standard deviation of 2 to 4 mV for most HICANNs on the BrainScaleS System, but with irregular peaks with an amplitude of several ten mV, which is much larger than the smallest PSP we want to detect. Compared to the Demonstrator Setups and the first prototypes BrainScaleS Systems, the noise levels improved noticeably.

We improve the signal by averaging over multiple PSPs taken in a single recording. However, the reduction in the noise level is not exactly by a factor of $\sqrt{N}$ for N averaged PSPs as we would expect from pure Gaussian noise. We assume that other noise source couple into the wire from HICANN to AnaRM. This can be caused by various sources in the HICANN itself as well as from interference from other

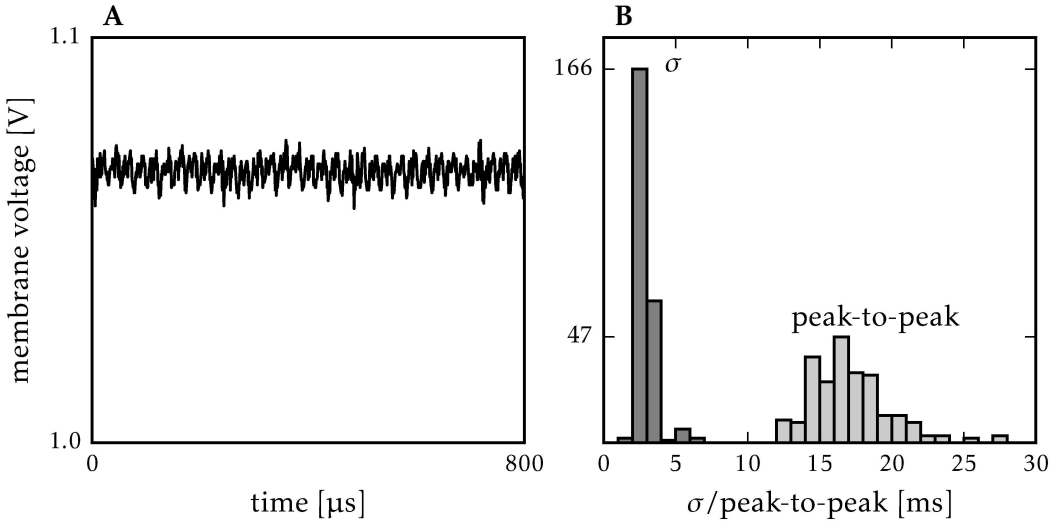


Figure 4.1 A Exemplary membrane trace. **B** Standard deviation σ and the differences of the highest to the lowest value (peak-to-peak) of traces from 240 HICANNs on Wafer 37 with module W_F01. Each trace has 960 samples. The data was collected with the test present in Section 3.3.3.

components of the system. The total length of the signal pathway is about 2 m, passing through the whole system. Lastly, AnaRM has no anti-aliasing low-pass filter in the input stage, which removes frequency components that are too high to be resolved by the ADC. A further improvement is expected by the next generation of AnaRM which is currently being developed by Joscha Ilmberger. It will be completely integrated into the BrainScaleS System and reduce the length of the readout-line to about a third.

Averaging the recorded PSPs greatly helps to cope with the noise. We use regular spikes from the background generators on HICANN as spike stimulus and record the membrane with AnaRM. The frequency of the spike stimulus needs to be small enough, so that the membrane can decay back to its steady state, so that the single PSP do not interfere with each other. On the other hand it should not be too low becaus measuring the membrane and transfering the data takes up the largest fraction of the time required for calibrations using this method [Mauch 2016]. Then we can cut the trace in equally sized slices each containing one PSP. We have to take care to locate the slices correctly, because we record asynchronously and therefore do have time-stamps that are synchronized with the HICANN clock which generates the spike stimulus.

Both HICANN and AnaRM work with independent clocks. For calculating the length of a recorded PSP we need the ratio of the PLL of $f_{\text{pll}} = 100\,\text{MHz}$ on HICANN and the sampling rate of AnaRM, which is $f_{\text{adc}} = 96\,\text{MHz}$. However, both rates have slight variations from their nominal values, which we take into account by adding

a correction to sampling rate $f_{\text{adc}}^{\text{corr}}$. We can then calculate the start m_i of the i-th chunk by

$$m_i = i \cdot f_{\text{stim}} \cdot \frac{f_{\text{pll}}}{\left(f_{\text{adc}} + f_{\text{adc}}^{\text{corr}}\right)}, \tag{4.7}$$

where f_{stim} is the frequency of the stimulus.

The required correction $f_{\text{adc}}^{\text{corr}}$ depends on the used combination of HICANN and AnaRM. We therefore developed a reliable method to measure $f_{\text{adc}}^{\text{corr}}$ using the preout-debug signal available on the first two synapse drivers from the chip. The debug signal allows to monitor the digital pulse from the synapse driver into the synapse array, which activates the synapse. When stimulated with the background generators, this is a very good source of a regular signal on HICANN, which we can use as reference to determine $f_{\text{adc}}^{\text{corr}}$.

The digital pulses have a height of 1.8 V and are one clock cycle long, but through the readout line it becomes slightly distorted. Therefore, the recorded peaks spread cover several samples, but the maximum is still over 1 V high. We obtain the location of each pulse by taking the weighted mean of the time-stamps, where we use the height of the pulse in a sample as weight.

We find usually values of $f_{\text{adc}}^{\text{corr}}$ of several hundred Hz, but we did not conduct a systematic evaluation of this. This measurement takes only several seconds including the configuration time. It can be neglected compared to the required time for the calibration that require averaging. Therefore we simply execute it once before each calibration. Investigating the stability of this value in time and under changing conditions remains to be done.

We can further reduce the influence of interference by choosing a period lengths, that are a number of prime number of PLL cycles. This avoids that regular events on HICANN show up in the averaged PSP.

The importance of applying the correction is shown in Figure 4.2. It shows that when averaging over 399 PSPs and ignoring the correction values, a shift of almost 1 µs occurs between the first and the last chunk. Even if the shape wrongly averaged PSP is not visibly impaired, we can detect this with the methods shown in the next section.

4.3.2 Fitting the Postsynaptic Potential

We now have a robust method to average over multiple PSPs in a single recording. We then can fit Equation (2.13) onto the obtained averaged PSP. We use the Levenberg–Marquardt algorithm as provided by the LMFIT software package. With well-chosen starting parameters the fit always converges . We then normalize the result, so that is $\tau_1 < \tau_2$. Examples of such fits are shown in Figure 4.3.

As we need an automatic evaluation of the fit quality, we conduct a χ^2 test on each fit. The quality of this test strongly relies on a good error estimate, which we obtain from a reference measurement. We record the membrane for each neuron

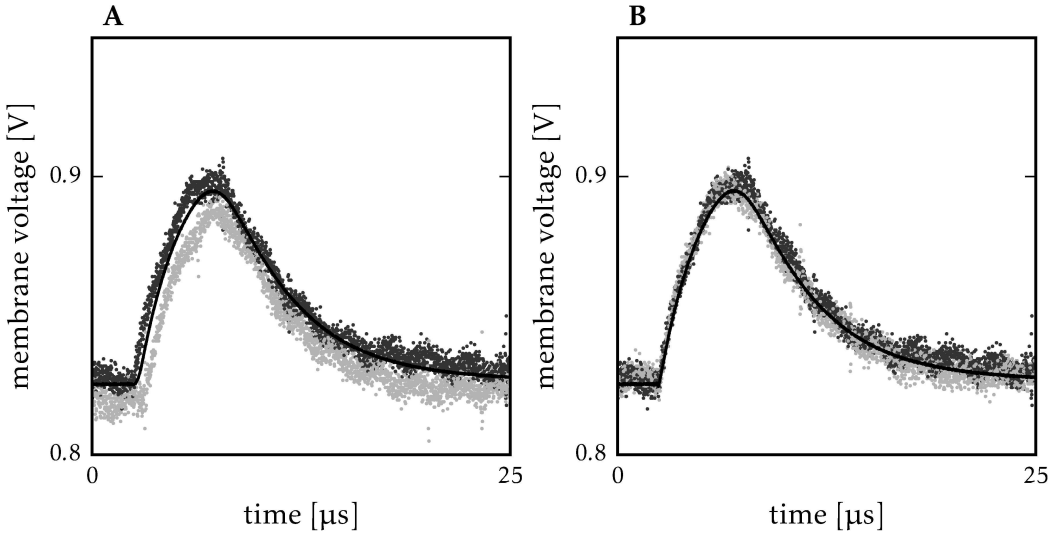


Figure 4.2 When averaging PSPs from regular spikes special care has to be taken to use the exact ratio between f_{pll} and the AnaRM sample rate f_{adc}. In both plots the first (dark dots) and the last (light dots) of 399 PSPs is shown. The averaged PSP is drawn as black line.
In **A** the nominal clock ration of $^{100}/_{96}$ was taken to determine the positions chunks for averaging.
In **B** the measured corretion term of $f_{\mathrm{adc}}^{\mathrm{corr}} = 604\,\mathrm{Hz}$ was used to calculate the start of the chunks, which is a typical value we found for AnaRM in the BrainScaleS Systems. The very small deviation has an significant influence as long traces are required to average a sufficent amount of PSP.

without spike stimulus. We use the same recording length as for the PSP recordings and apply the same averaging method. This avoids that non Gaussian noise effects disturb the estimate. The standard deviation of the averaged recording σ_{est} is then used as error estimate. We then can obtain reduced χ^2 for a fit x' on a trace x with n elements using

$$\chi^2_{\mathrm{red}} = \frac{1}{n - n_{\mathrm{p}}} \cdot \sqrt{\sum_{i=0}^{n} \frac{x_i - x'_i}{\sigma_{est}}}, \tag{4.8}$$

where n_{p} is the number of free parameters of the fitted function.

The χ^2 test allows us to reliably detect, when model and data no longer match. This can happen for various reasons. In HICANN revision 2, saturation effects can strongly impair the shape of the PSP. Also for to large PSP the model fails, because we leave the linear range of the neuron components. Lastly, unforeseen effects, like strong signals coupling into the readout line, did impair single measurements. This

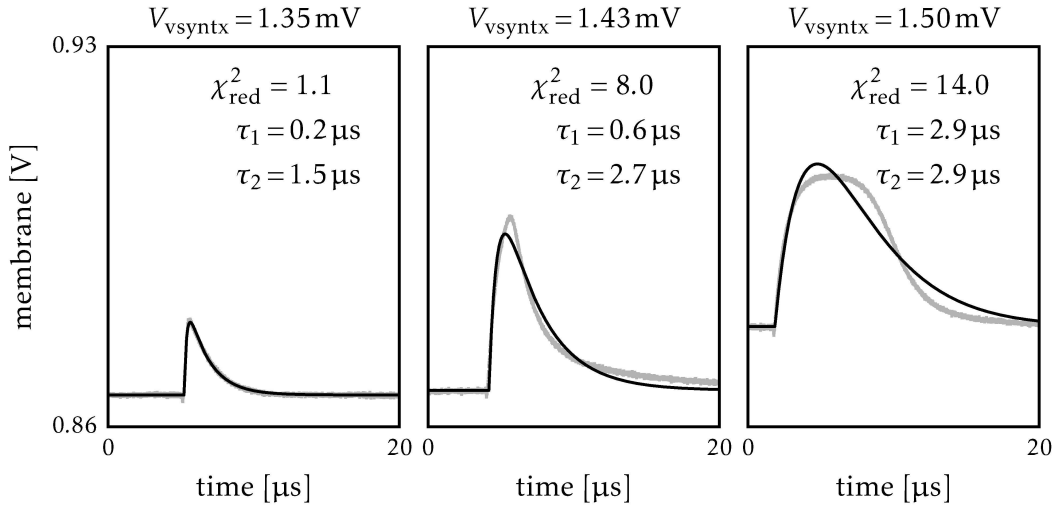


Figure 4.3 Exemplary fit of PSPs for three different settings of V_{syntc}. On the average of 399 PSP (gray) Equation (2.13) was fitted (black). The time-constants and the reduced χ^2_{red} is annotated in the plot. The first PSP shows a very good fit. Then the shape of the PSP deteriorates for larger V_{syntc}. For the highest V_{syntc} value we observe, that is $\tau_1 = \tau_2$. This happens for almost all badly deteriorated PSPs.
The three PSP were taken from the calibration of V_{syntc} on HICANN revision 2, as shown in Section 6.2.3.

problem no longer occurs with the improvements of the system.

We evaluate the method using the data from the measurements of HICANN revsion 2, which is shown in Section 6.2.3. For this we generated PSPs for 11 different settings of the control parameter of the synaptic time-constant V_{syntcx}, from which we obtained a total of 5632 fits on 512 neurons. As shown in Figure 4.4, most of the fits indeed are distributed closely around $\chi^2_{\text{red}} \approx 1$. The remaining fits have a $\chi^2_{\text{red}} > 10$, but only for larger V_{syntcx} values.

We further analyze how the χ^2_{red} value correlates with the height of the recorded PSPs. The result is shown in Figure 4.5. At low V_{syntcx} values only PSPs with $\chi^2_{\text{red}} \approx 1$ and heights up to 50 mV occur. This is not surprising because 50 mV still lie in the linear range of OTA_1, as seen in Figure 2.4. But there is also a significant proportion with a very low amplitude. That these also have $\chi^2_{\text{red}} \approx 1$, shows that the error estimate works well, because the model is also valid for zero sized PSPs. For larger V_{syntcx} values the correlation between height and χ^2_{red} increases. A closer inspection shows that this is caused by deformed PSP as shown in Figure 4.3. The data shows well that χ^2_{red} is suitable to estimate the quality of the fit. The implications of the effects shown here are discussed in Chapter 6.

We also observed that the used fit algorithm or the model itself have the tendency

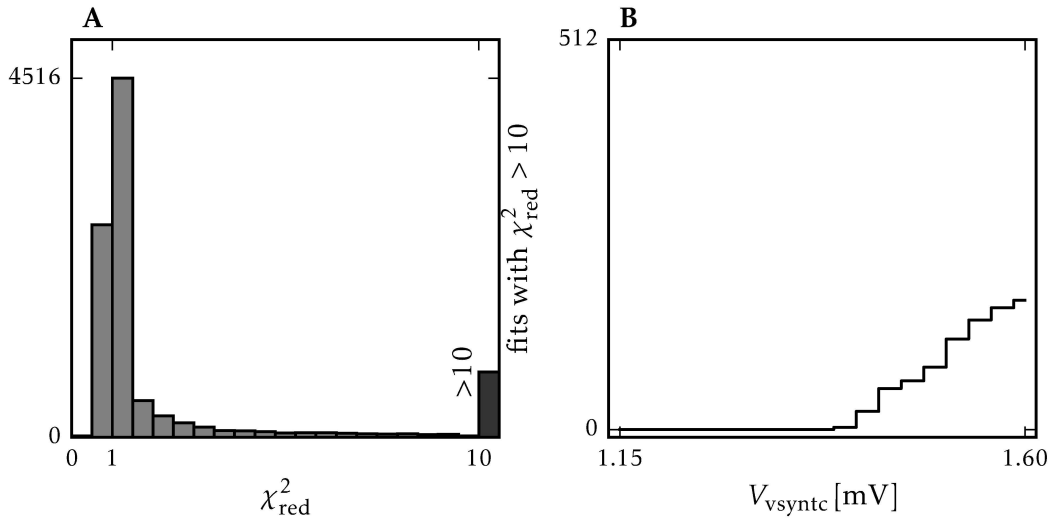


Figure 4.4 A Distributions of χ^2_{red} for PSP-fits of 19 measurements with V_{syntc} from 1.15 to 1.60 mV. The dark bar includes all fits with $\chi^2_{red} > 10$. It shows, that the model describes most PSPs well. **B** shows that most bad fits occur for very large V_{syntcx} values. This is caused by distorted PSP-shapes, as it can be seen in Figure 4.3.

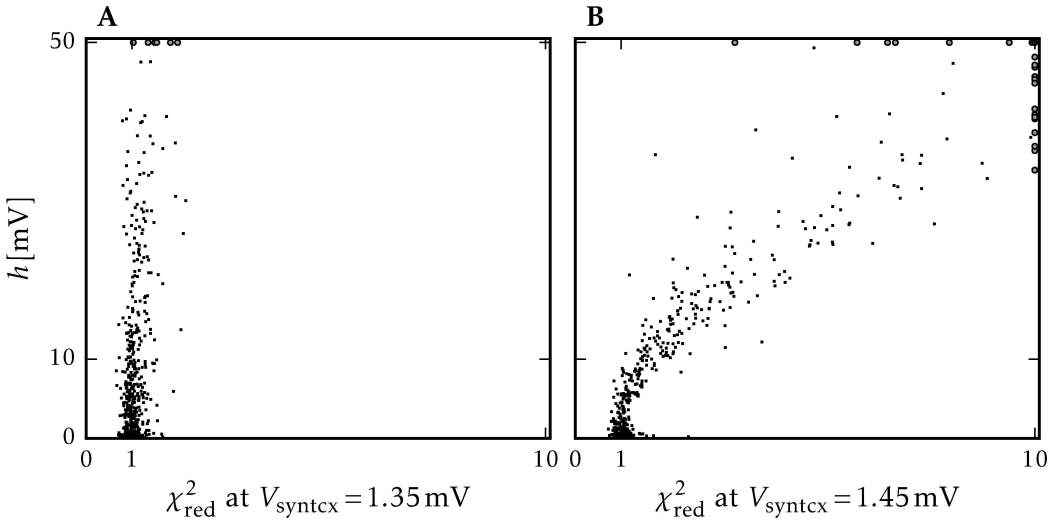


Figure 4.5 Correlations between χ^2_{red} and the fitted PSP-height h for different values of V_{syntcx}. The gray dots are cropped to the range of the plot. For low V_{syntc} the PSP fit very well to the model, even for quite large heights. For larger V_{syntc} the shape of the PSP quickly deteriorates and the model doesn't fit well. This is reliably detected by the χ^2_{red} test.

to prefer the case $\tau_1 = \tau_2$, which is a special case in Equation (2.13). This effect will be more immanent for the evaluation of the results of HICANN revision 4.1. We did not find a suitable solution to avoid this effect so far, but tried several other fitting algorithm without success.

The method also works for inhibitory PSPs, even if the shown examples so far only were only excitatory PSPs. The presented method is well suited to analyze PSP on HICANN. However, it requires quite long measurement times to obtain PSP with sufficiently low noise. This concludes our measurement methods for HICANN.

4.4 Transitor-Level Simulations of Circuits

Aside from measurements we also conducted transistor level simulations of the whole neurons circuits or its components. This way we can get insight into the circuits, that are not directly accessible for measurements. As a simulator we used the Spectre Circuit Simulator in combination with a Berkeley Short-channel IGFET Model [Cheng et al. 1996] (BSIM), which are provided by UMC [UMC 2004], the manufacture of HICANN, and described in the SPICE format [Quarles 1989]. The BSIM models are accurate models for the used transistors, which can also include data about mismatch effects and process variations. Therefore these models are non-public and the author is required to sign an Non Disclosure Agreement (NDA) to legitimately use them. We therefore cannot provide further details on the used model.

In the following we will first describe how the simulator is used. Afterwards we present the simulation setups we created. Each is optimized for the specific tasks to reduce the simulation time and simplify the simulation setup. Lastly we created an integration of a full neuron circuit simulation into the calibration tool.

4.4.1 Simulation Methods

We used the Spectre Circuit Simulator with a small Python wrapper, written by Sebastian Billaudelle. This wrapper replaces the graphical circuit design tools by issuing the netlist extraction and setting up the simulator parameters and analyses to be performed by Spectre Circuit Simulator. Using Python has the advantage, that we could reuse the analysis tools already created for measurements and integrate the circuit simulations into the calibration frame work.

The Spectre Circuit Simulator supports a wide range of analyses. Here we used DC-simulations, which determine the steady state of a circuit for a given set of input parameters, and transient simulations, which simulate the temporal evolution of voltages and current in a circuit for changing inputs. The former is usually used to characterize the properties of single sub-circuits, while we use the latter to investigate the dynamics of the synapses and synaptic inputs, which can not be directly measured.

The average circuit behavior is described by the typical case parameters in the BSIM provided by UMC. We use this to simulate the nominal behavior of the circuit

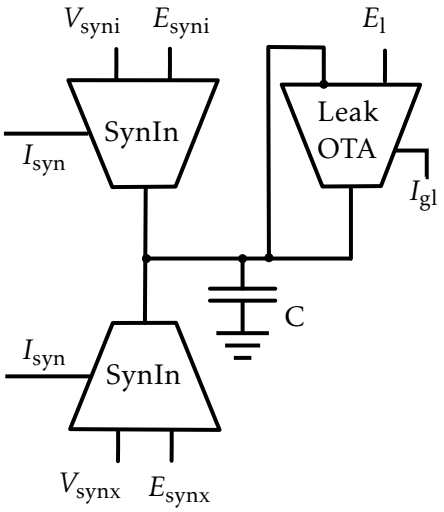


Figure 4.6 Schematic of the Simplified Neuron Simulation. This simulation contains only the most essential parts of the neuron to study the interaction between the synaptic input and the leakage conductance.

in general matches with the neuron model. Additionally data about mismatch is included in the BSIM provided by UMC . We use these in Monte Carlo simulations, where the model parameters are randomized simulating mismatch effects, as they occur in HICANN. Note that we do not have mismatch data for all used transistors used in revision 4 and 4.1. We executed all simulations with the conservative error preset of the simulator and a nominal temperature of 50 °C. The default parameters for the presented Simulations are listed in Appendix A.5.8.

4.4.2 Simplified Neuron Simulation

We created a strongly reduced simulation of the neuron by removing all components except the leakage term and the synaptic inputs. This simulation allows us to focus on the essential interaction between the synaptic inputs and the leakage conductance. It increases the simulation speed compared to a full neuron circuit simulation, which allowed us to take a larger number of Monte Carlo samples. Also the number of required parameters is largely reduced. A schematic of the simulated circuit is shown in Figure 4.6, where also all input parameters are shown. We use this simulations only for revision 2 of HICANN to obtain the results shown in Chapter 6.

The simulation has the regular neuron parameters E_{l}, E_{syni}, E_{synx}, V_{syni}, V_{synx} and I_{gl} as parameters. These are realized as ideal voltage and current sources. Synaptic events are simulated by sending a rectangular current pulses of height I_{syn} and length of 10 ns matching the PLL of 100 MHz

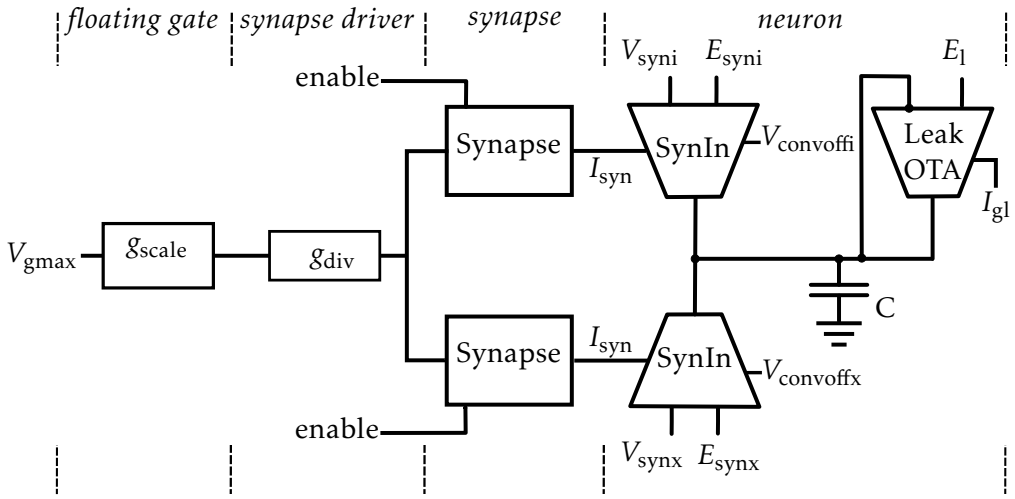


Figure 4.7 Schematic of the Simplified Neuron Simulation with Synapses. We added the complete analog circuits involved in the generation of the synaptic events to the simulation.

4.4.3 Simplified Neuron Simulation with Synapses

We also simulate the neuron in interaction with the synapse circuit and extend the simplified neuron simulation for this. We add the complete analog circuits involved in the generation of synaptic pulses, including those from the scaling of V_{gmax} in the analog parameter storage, the synapse driver and two pairs of synapses. We also simulate the quite large parasitic capacity from the line connecting the synaptic input to all 220 synapses (or 224 in HICANN revision 2). For this a capacity of 1 pF added to it, where value was extracted from the chip layout. A simplified schematic of the simulation is shown in Figure 4.7.

We created a version of the simulation for revision 2 as well as 4.1. Additionally to the parameters of the simplified neuron simulation, we added the synaptic reference current V_{gmax}, the scaling factor g_{div} and the synaptic weights w of the four synapses. Additionally we added $V_{convoffi}$ and $V_{convoffi}$ neuron parameters for revision 4.1. We set the states of the SRAMs for g_{div} and synaptic weights w via simulations initial conditions. The synapses are triggered by pulling up the enable line of the synapse for one PLL clock cycle (10 ns). We follow up with functional details of the circuits in Section 5.3.

4.4.4 Complete Neuron Simulation for the Calibration Framework

Lastly we create a simulation of the complete neuron and integrated it into the calibration framework[1]. This work was done in cooperation with Mitja Kleider and Paul Müller. Licensing issues required us to split the simulation into two parts.

[1] It is not related to the HALbe backend for simulation of analog circuits [Müller 2014]

The simulation core runs on dedicated machines for circuit simulations, while the calibration tool can be executed on an arbitrary other machine.

The simulation itself is based of the one used by Millner [2012]. It consist of two complete neuron circuits, that can also be interconnected. All voltage and current parameters are provided by ideal current and voltage sources, so that floating gates effects aren't included. Spike input is given as a current pulse, like in the simplified neuron simulation. The simulation included the small and large membrane capacity and the scaling factors applied to various currents as described by Millner [2012]. Lastly we added a VerilogA module that can stop the simulation after a certain number of spikes. This can reduce the simulation time drastically, because the spiking process is computationally very costly.

The simulation needs an initialization phase. In this phase the SRAMs for the digital neuron configurations are programmed by emulating the controller pulses. Also membrane and adaption circuits are shorted to ensure that the calculation of the initial state in the simulation converges. Afterwards we implemented a configurable settling phase, so that the neuron can adjust to a new steady state after these steps are done. The data recorded during the settling phase is discard. This allows the returned data to be handled the same way as actual membrane recordings.

In the calibration framework itself we only needed a few changes. The process of creating measurement steps is normally done as resulting in a StHAL container holding the complete configuration. Afterwards we don't connect to the system and skip the basic configuration and continue with the measurement that steps over all neurons. Here we don't readout the membrane, but instead extract the configuration data from the given neuron and its possible spike input from the StHAL container and send it to the simulation server. The recorded traces are then returned to the calibration framework. These contain the membrane voltage, but can also contain any voltage or current that can be recorded. The simulator uses for the integration adaptive step sizes, but some calibrations rely on the regular spaced samples. We therefore resample the simulation data is to match the sample rate of AnaRM, using a linear interpolation. Optionally the framework will cache the results for now simulation setting on disk. This way we don't need to repeat simulations that were already done, when we are working on the analysis of the data.

The full power of this integration shows when using it with Monte Carlo simulations. In this setting a random seed is given and each neuron is mapped to a Monte Carlo sample. This way we always obtain the same variations for a neuron in the simulation. We calibrate and evaluate them, exactly like the neurons in the real system. With this approach we could develop the prototypes of the calibrations for revision 4 already during the design and production phase of the new chips.

4.5 Summary

In this chapter we first define our requirement for calibrations. A clear definition of the transformation function is developed and the process to obtain the parameters formalized. We then describe the usage of the BrainScaleS System and the details of the measurement process. As we cannot measure the time-constants in the synaptic input directly, we developed an analysis depended on the PSP of a synaptic input. This requires a precise averaging method which compensates the different clock domains of HICANN and AnaRM. Lastly we created simulations for parts of HICANN that can not be measured directly. This also includes a integration of transistor-level neuron simulation into the calibration framework. With these methods and tools we are able to conduct the characterization and calibration of the synaptic input and the synapses.

5 General Measurements and Simulations

In this chapter we present three topics, that set the base for the calibration of the synaptic inputs. In Section 5.1, we begin with a characterization of the analog parameter storage. As discussed in Section 4.1, the precision, which is obtained for the single parameters, directly affects the maximal possible calibration result. We present here a new view on the cause of the trial-to-trial variability of the single floating gate cell. While improving the precision was outside of the scope of this thesis, the results show a new possible approach to reduce the variability.

In section Section 5.2 we repeat the known calibration for the resting, reset and threshold potential for HICANN revision 4.1. The potentials are required for the calibration of the synaptic input. As none of these circuits were changed, we only needed minimal changes to the calibration routines. We find that previous results can be well reproduced and that we reach a trial-to-trial variability that matches well with the one we found for the floating gates.

At last in Section 5.3, we present a model to estimate the strength of synaptic events and their effect on the neuron. We here use circuit simulation as base for our results, because these circuits are not directly accessible for measurements. We find that the strength of the synapses has an offset, that depends only on the chosen synaptic weight. We can quantify this offset and provide a model to predict the strength of synaptic events. Afterwards we can show that the observed effects can be also observed on the neurons.

5.1 Precision of Analog Neuron Parameter

The analog neuron parameters lay the base for the maximal achievable precision of the neuron calibration. We use for the measurements conducted for this thesis the default parameters for the controller as they are provided by StHAL. However, these are not specifically optimized for the wafer systems we are using. So we first evaluate the precision we can achieve using these parameters. Afterwards, we show examples how the different parameters of the floating gate controller affect the programming and how differently the single floating gate cell react to the controller. We argue, that combination of both makes it difficult to optimize the trial-to-trial variations of the parameters in HICANN.

5.1.1 Currently Achieved Precision

We can evaluate the precision by directly reading the output from the floating gate cells. For the programming, the output of each cell can be connected to the

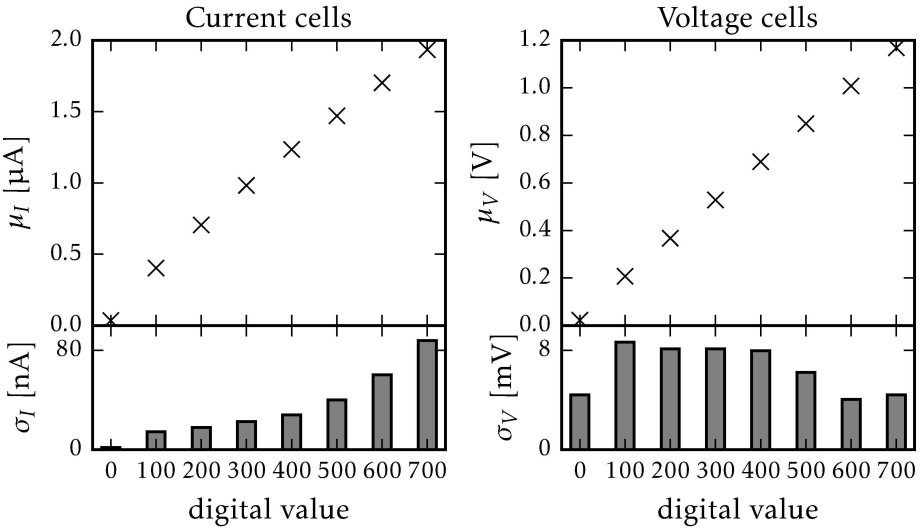


Figure 5.1 Precision of the floating gate cells. For this measurement all cells of a HICANN were programmed to the same digital target value (DAC). For each value $N = 100$ measurements were done. The result for current and voltage cells are shown in different plots. In both plot, the upper half shows the reached mean value and the lower half the standard deviation with higher resolution. The readout buffer of HICANN saturates at voltages above 1.4 V, there not the whole range of possible settings was measured.

comparator in each floating gate block. From there a readout line can be used to read any cell in a floating gate block. The output of the current cells is also converted to a voltage for comparison and therefore also read out as voltage. As stated in Hartel [2016], we obtain the actual current of the parameter by

$$I_{\text{out}} = \frac{1}{4} \cdot \frac{1}{150\,\text{k}\Omega} \cdot V_{\text{out}}. \tag{5.1}$$

We further need to note two limitations of this measurement method: Firstly, as described in Section 2.2.4, the current towards the comperator and the current towards the neuron are generated by two different current mirrors. The ratio between those two is subject to further mismatch. Therefore the actual value parameter value, as seen by the neuron, deviates from the value we can measure. Secondly, the readout buffer on HICANN limits the read voltages to about 1.4 to 1.6 V. This sets an upper limit to the parameter values we can measure.

Figure 5.1 shows the precision we can reach with the current settings of the controller. For the voltage cells the relation between digital control value and the resulting current of voltage is linear. The current cells show a slight deviation from a linear course. The maximal precision we can reach varies between cells types and target values. For the current cells we reach a standard deviation $\sigma_\text{I} = 5$ to 80 nA

and for voltage cells $\sigma_V = 4$ to 8 mV. For current cells the error rises with larger values. For voltage cells it is best for digital values between 600 and 700, which corresponds to voltages larger than 1.0 V. The currently known best results lead to variation below 4 mV for all voltage cells [Kononov 2011].

To the authors knowledge, the currently used set of parameters was manually tuned for a specific system. Due to the various parameters of the controller as well as the requirement to provide a fast programming scheme, this is a time-consuming process. For comparison the current scheme needs about 15 s to program all floating gates on a single HICANN. While trying to improve these results, we noticed that this was a more difficult task than expected and out of the scope for this thesis. However, we will still present our intermediate results as they might show a new approach towards improving the programming scheme of the floating gate controller.

5.1.2 Variation in the Programming of Cells

The problem in finding optimal parameters for the floating gates controller lies in the inhomogenous response of the different cells to the programming pulses of the controller. We measured for each cell on a single HICANN 100 trials at various digital values and calculated the standard deviation of the measured currents and voltages for each of those cells separately. We excluded the rows 1, 3 and 17 from the measurement and keep them at zero, because these contain the I_{conv} parameters. We use various settings of the digital control parameter of 200, 300, 400, 500, 600 and 700.

The results are shown in Figure 5.2. We notice that the reached precision varies greatly between the different cells. For the current cells, we observe a quite broad distribution. At a digital value of 200, it spreads from near zero to 30 nA, while at 700 this spread increases to 10 to 80 nA. For the voltage cells the opposite shows: for a digital value of 200, the distributions spreads from 5 to 10 mV, while at 700 it only spreads from 1 to 4 mV.

We can offer an explanation for the behavior, if we look at the charging process of the cells. Similar measurements were priviously conducted only on for single cells on test chips [Hock 2009] and on the MCC [Hartel 2016]. Using unbuffered configuration, the charging cannot be observed directly. But we can run the controller multiple times and record the cells after each run. For this, we need to set a large target value and set the *maxcycle* parameter to a small number, while disabling the *acceleratorstep* of the controller. We then can program and read out the cell repeatedly until the cell reaches its maximum value. We effectively average over multiple programming cycles by using a *maxcycle* > 1. A larger *maxcycle* setting speeds up the measurement significantly, but reduces the resolution. We can also measure the discharging process, using this method.

We recorded this process for two settings for VDD12 = 10.8 and 11.2 V and two different *writetime* settings of 5 and 50 with *maxcycle* of 30 and 5 respectively. The other parameters are set to *pulselength* = 1, *acceleratorstep* = 63 and *readtime* = 63. The charging curves are shown in Figure 5.3. The strong effect of the programming

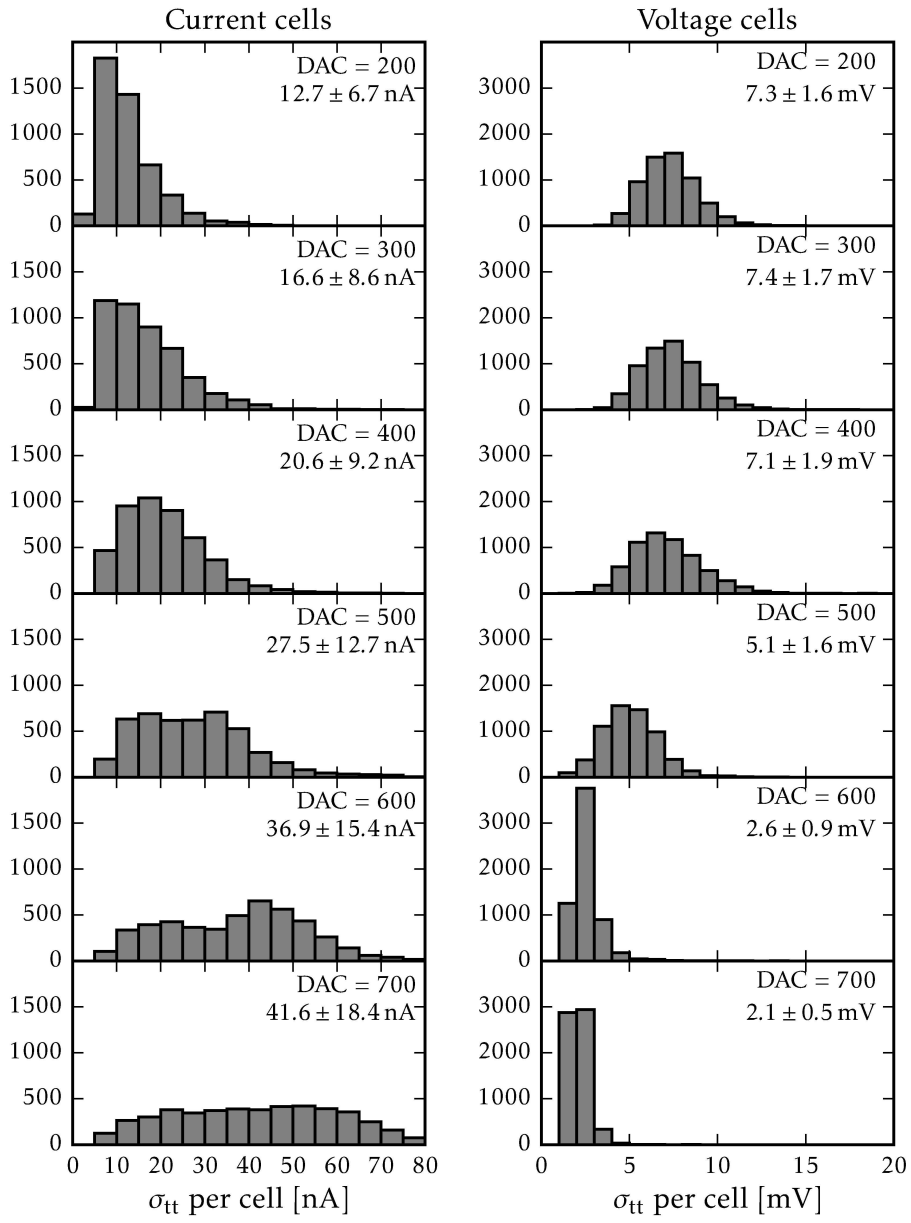


Figure 5.2 Programming precision per cell dependent on the target value. All floating gate cells of a single HICANN were programmed to the same target value. Only the rows 1, 3 and 17 were kept at zero, because these contain the I_{conv} parameters. For each cell the standard deviation of $N = 100$ trials is added to the histogram. We observe that the reachable precision varies strongly between the single cells and between the different target parameters.

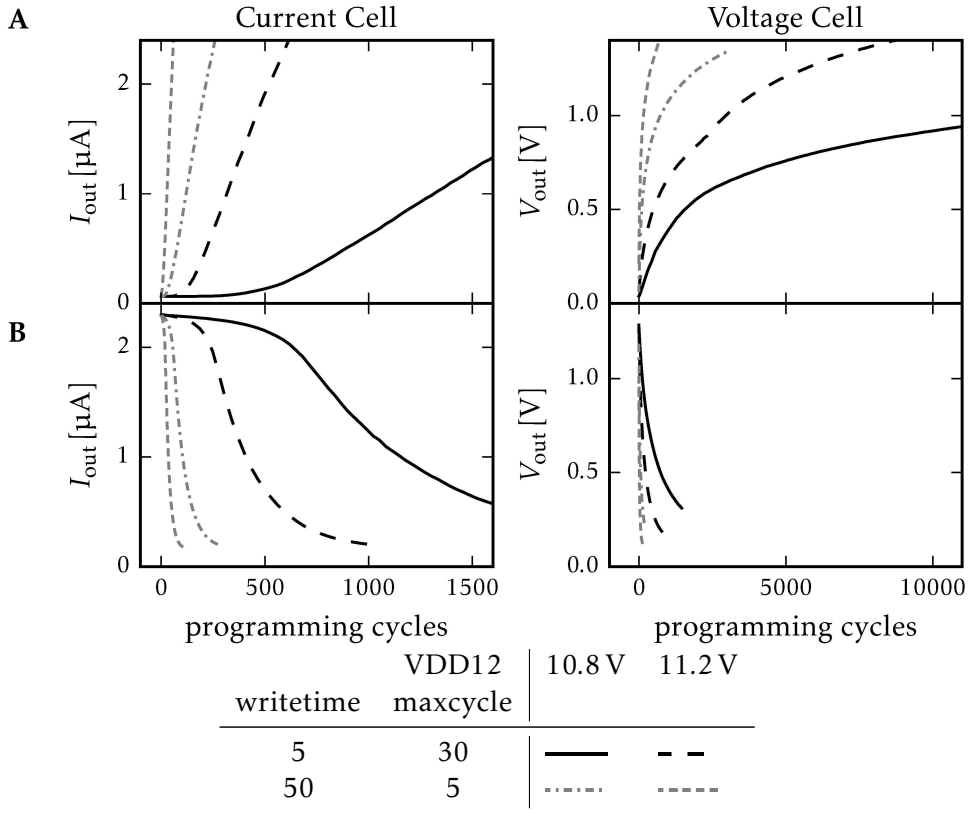


Figure 5.3 Required programming cycles to **A** charge a floating gate cell **B** discharge it to a certain value. Here two exemplary cells for multiple parameters are shown, as we see later the characteristics vary between the cells. The cells start at their minimal value (or maximal value for discharging). Then the controller is set to run a fixed number of programming cycles towards the maximal value (minimal for discharging). Then the controller is started and afterwards the values cells are read. This process is repeated until the floating gates reached at least 1.2 V (0.2 V for discharging). Hereby the maxcycle parameter is set to 5 for a writetime of 50 and to 30 for a writetime of 5.

voltage VDD12 is caused by the exponential dependency of tunneling current on the potential [Loock 2006]. Further, a larger *writetime* causes a longer programming pulse and the charging or discharging becomes more effective. Further the charging and discharging effect of each programming cycle is strongly dependent on the current output value of the floating gate. While the response of the current cells is linear for most of the output value range, the charging of the voltage cells becomes less effective the larger the output voltage already is. The difference between current and voltage cells is caused by the different design of the cells. The source and drain potentials of the floating gate transistor are both higher in the voltage cell. Further the current charge of the cell has an influence to the potential difference created to

induce the tunneling currents. Lastly, we notice that discharging the voltage cells is substantially faster, than charging them.

This characteristics influence the possible precision the controller can achieve. As the controller will always conduct a full programming cycle with the given *writetime*, it can easily shoot over the target for to large *writetime* settings. On the other hand, a short *writetime* requires a very large number of programming cycles: for the cell shown in Figure 5.3 about 25 000 cycles are needed to go from 0.0 to 1.2 V. Further we notice that, at least for the voltage cells, too short programming pulses no longer have measurable effect on the cells. The right trade-off is essential here. The strong dependency on the current charge of the cell complicates the selection of the best parameters further. While longer pulses are needed for lower voltages, these are bad at high voltages. In contrast the controller stops at a given limit of pulses, so that short pulses will not be able to reach the required precision for low target voltages.

This difference is already reflected in the two-pass programming scheme currently used in StHAL. Here at first the parameters are written with few large pulses coarsely over their target value and afterwards a second programming pass programs once down and once upwards with shorter pulses.

The difference also affects the programming of different cells. We analyzed the charging curves of 129 cells in each one row of current and voltage cell. We use VDD12 = 10.8 V and a *writetime* of 5, because this is the default supply voltage and the *writetime* parameter used for the second programming passes of the controller. We use a linear approximation, where we fit a linear function to all values of curve in a given range of the output parameter. For the current cells we fitted to the range between 0.66 and 2.0 nA and for the voltage cells at two ranges between 0.45 and 0.55 V and between 0.75 and 0.85 V. Figure 5.4 shows for two curves how the linear approximation is done and the distribution resulting slopes of the charging curves at this points. The current cells show a response of $1.9 \pm 0.8\,\mathrm{nA\,cycle^{-1}}$ and the voltage cells at an output value of 0.5 V a of $0.05 \pm 0.04\,\mathrm{mV\,cycle^{-1}}$ and at an output value of 0.8 V a slope of 0.16 ± 0.13 nA.

We assume that this difference between cells are also caused by mismatch. A compensation for this differences is difficult with the current controller, because the cell of a row are always programmed at the same time. As most cells in a single row are programmed to the same or at least similar values, a controller scheme designed for a certain target value might greatly improve the overall performance here in precision as well as in speed. However, further studies remain to be done here. We hope, that the results presented here show new approaches to improving the precision of the floating gates.

5.1.3 Summary

We have shown, that currently reached precision is below the so far best known [Kononov 2011]. This sets an upper limit to the possible precision the shown calibration methods can reach. We can observe this directly for the calibration of the neuron potentials, shown in Section 5.2.

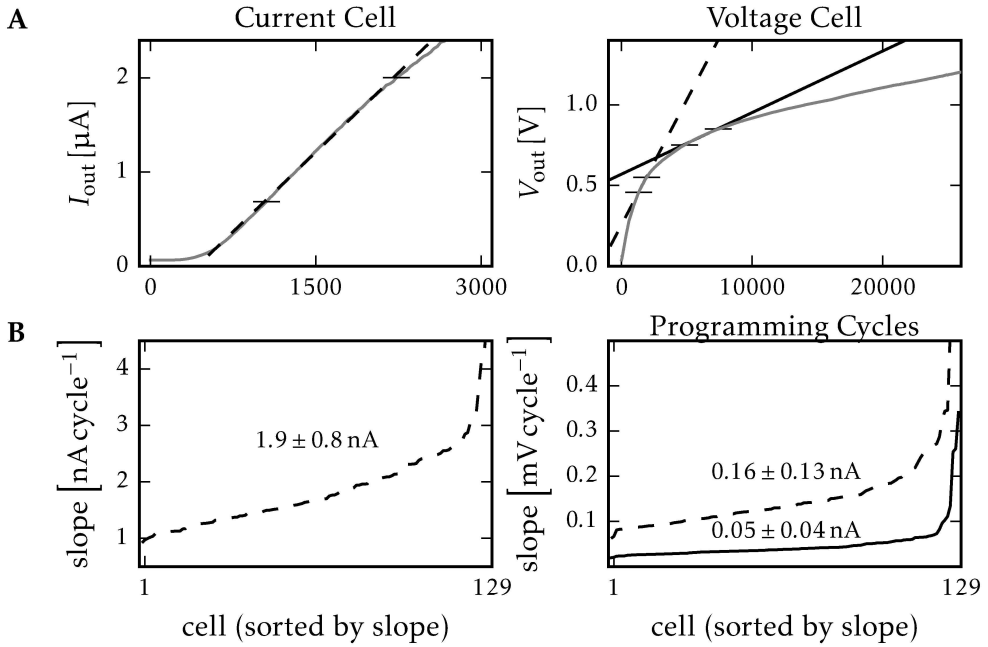


Figure 5.4 Response of the floating gate cell to the programming pulses. **A** We fit a linear function to the data to estimate the slope of the curve. For the current cell we fit to the data between 0.66 and 2.0 nA and for the voltage cells we fit to two regions: firstly between 0.45 and 0.55 nA and secondly between 0.75 and 0.85 nA. **B** Charging rate obtained from the fit for all 129 cells of one current and one voltage row. We see that the cells respond very differently. For the current cells the range is from 0.9 to 6.0 nA. For the voltages cells in the lower region in the range from 0.06 to 1.03 mV and in the upper region from 0.02 to 0.34 mV. The inhomogeneity of the cell response makes it difficult to find ideal settings for the floating gate controller.

Further the trial-to-trial variability of the single floating gate cells varies itself between the different cells as well as for different target values. We propose, that this is the reason for the difficulties to find a generally applicable programming parameter scheme for the floating gate controller, because each cell responds differently to the programming pulses of the controller. The method we shows could be extended to create a general model of the response of the floating gate cell, dependent on the controller parameters and the supply voltage VDD12. Such a model can be used to optimize the parameters and the programming scheme offline avoiding the lengthy programming times. Furthermore this model could be used in the design a new controller for the analog parameter storage, that handles the variation of the different cells better.

5.2 Calibration of the Neuron Potentials

A perquisite for the calibration of the synaptic input is the calibration of the synaptic potential. The base for these calibrations was already done by Schwartz [2013] and the methods have been improved and integrated into the calibration framework by Schmidt [2014], where results for HICANN revision 2 can be found. The transfer of the method toward HICANN revision 4 and 4.1 was straight forward and the calibration leads to good results. However minor changes were required, which we will cover in the following. We now want to shortly summarize the methods and present the results. Additionally to the potentials, the readout bias of the neurons is calibrated as described by Schmidt [2014], but the results are not shown here.

The deviation for the potentials are mostly caused by DC-offsets of active components like the threshold cooperator or the leakage OTA. These can be very good corrected using a linear transformation function. Usually 3 to 5 measurements lead already to sufficiently good results.

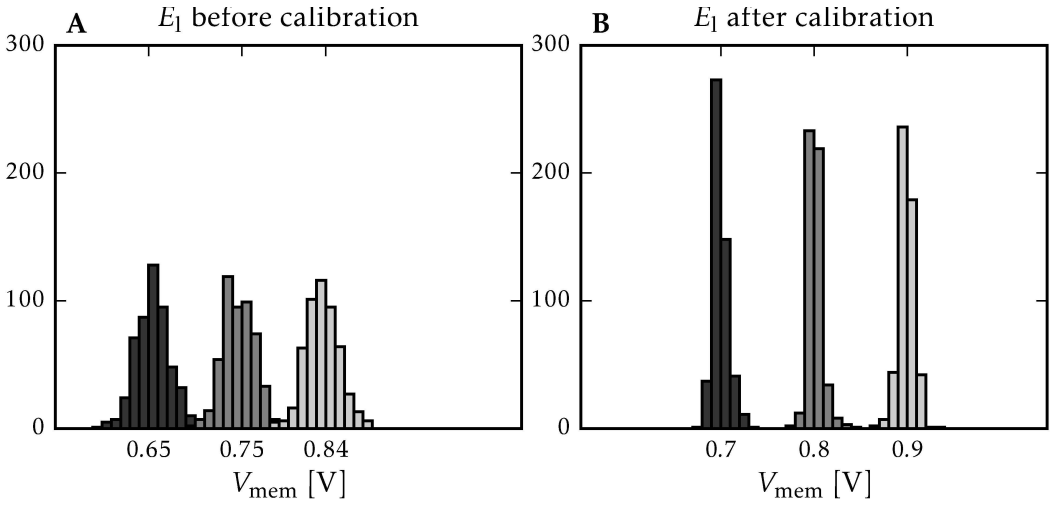
5.2.1 Resting Potential

The resting potential is calibrated by leaving the neuron in its resting state and measure the mean membrane voltage. We hereby now disable the synaptic inputs again, as originally done by [Schwartz 2013] by setting I_{conv} to zero. The result is shown in Figure 5.5, the parameters can be found in Appendix A.5. For the evaluation step, the inputs are enabled again and the complete calibration for the bias generator is applied, which we present later in Section 7.2. As we see, we can now be well calibrated towards a precision of about 8 mV, This is indeed the maximal trial-to-trial variability the voltage parameters have for target values around 0.8 V with the currently used settings of the floating gate controller.

5.2.2 Reset and Threshold Potential

The calibration methods for the reset V_{reset} and threshold potential V_{t} remain unchanged. The resting potential is set above the spiking threshold to bring the neuron in a state of permanent spiking. Details of the analysis of the traces for both methods are documented in [Schmidt 2014]. The threshold is found taking the average of the local maximal of the recorded trace $\langle V_{\text{peak}} \rangle$. The obtained precision is significantly trial to trial variation is significantly smaller than for the other two parameters because the target values for this calibration lay in the sweet spot of the trial-to-trial variability of the voltage parameter for the use floating gate controller settings, as shown in Section 5.1.

The reset potential is measured using the base line V_{base} during the refractory periods between the spikes. It is maximized to increase the effect. The result is shown in Figure 5.7 and we listed the used parameters in Appendix A.5. Also the reset potential could be perfectly reproduced on HICANN revision 4. The error we obtain is also dominated by the trial-to-trial variations.



	A before calibration		**B** after calibration	
	target[V]	$\mu \pm \sigma$[V]	target[V]	$\mu \pm \sigma$[V]
	0.7	0.655 ± 0.018	0.7	0.700 ± 0.008
	0.8	0.747 ± 0.018	0.8	0.801 ± 0.007
	0.9	0.836 ± 0.018	0.9	0.899 ± 0.008

Figure 5.5 Distribution of the resting potential before and after calibration on HICANN revision 4.1. Bin-size: 10 mV.

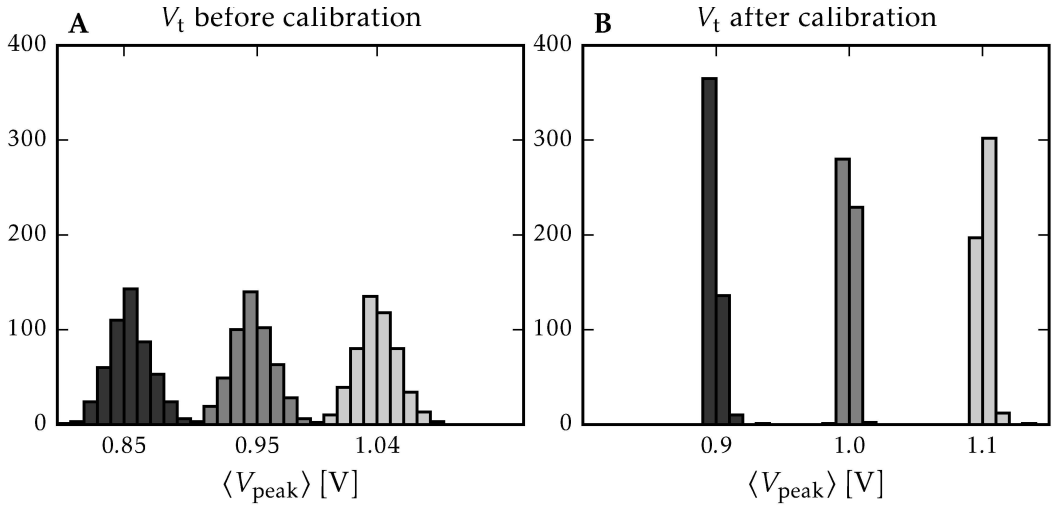
5.2.3 Summary

We here only presented the absolute minimum of the calibrations required for the calibration of the synaptic inputs following now first for revision 2 and then for revision 4.1 . The calibration of the AdEx parameters are subject to the ongoing work of Mitja Kleider. The results obtained here perfectly reproduce the work of Schwartz [2013] and Schmidt [2014] and show that the revision 4.1 shows the same performance here as the previous revisions.

5.3 Simulation of Synaptic Weights

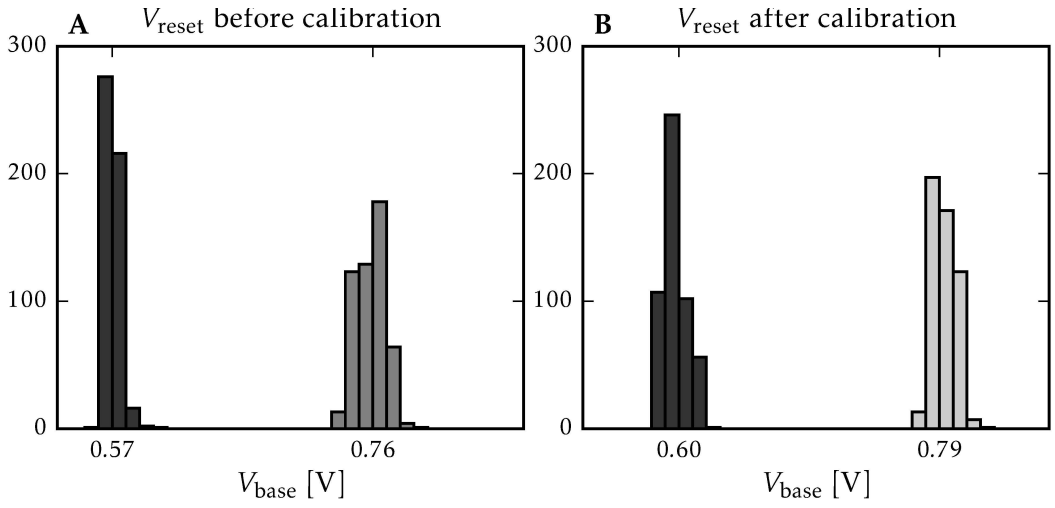
The characterization of the calibration of the synaptic input relies on synaptic events as stimulus. While in the AdEx model, as described by Equations (2.3) and (2.4), the weight directly corresponds to the rise in the synaptic conductance, this is a more complex process in the hardware. As described in Sections 2.2.1 and 2.2.3, first a current pulse I_{syn} is generated in the synapses, which is then integrated by the synaptic input of the neuron. In this section we focus on the generation of the current pulse. The integration and decay are described for HICANN revision 2 in Section 6.1.2 and for HICANN revision 4.1 in Section 7.1.2.

We first show a detailed description of the circuits. We then simulated these



	A before calibration		**B** after calibration	
	target[V]	$\mu \pm \sigma$[V]	target[V]	$\mu \pm \sigma$[V]
	0.9	0.855 ± 0.015	0.9	0.899 ± 0.004
	1.0	0.947 ± 0.015	1.0	1.000 ± 0.003
	1.1	1.040 ± 0.015	1.1	1.101 ± 0.004

Figure 5.6 Distribution of the threshold potentials before and after calibration on HICANN revision 4.1. We reach the trial-to-trial error of the floating gate cells in the voltage range as shown in Section 5.1.1. Bin-size: 10 mV.



	A before calibration		**B** after calibration	
	target[V]	$\mu \pm \sigma$[V]	target[V]	$\mu \pm \sigma$[V]
	0.6	0.570 ± 0.005	0.6	0.597 ± 0.008
	0.8	0.758 ± 0.010	0.8	0.793 ± 0.008

Figure 5.7 Distribution of the reset potentials before and after calibration on HICANN revision 4.1. Bin-size: 10 mV.

circuits for a wide range of parameters. Hereby an unexpected charging effect in the synaptic circuit was observed. We incorporate this effect into a model for the synaptic strength and successfully validate the model against the data. Lastly, we show that the expected variations due to mismatch are around 10 % and close with a short summary.

5.3.1 Synaptic Circuit

The amplitude current pulses I_{syn} of a synapse are controlled by three parameters: the reference V_{gmax} current, the current divider g_{div} in the synapse driver and the weight w of the synapse itself. The length of the pulse is controlled by the PLL. At 100 MHz, as we use it, each pulse has a width of $t_{\text{syn}} = 10\,\text{ns}$. The width can be modified by the STP, to realize facilitation or depression of the weights, but we consider here only the static use case. As described in Section 2.2.3, the nominal synaptic current of a pulse is

$$I_{\text{syn}} = V_{\text{gmax}} \cdot g_{\text{scale}} \cdot \frac{w}{g_{\text{div}}}. \tag{2.18}$$

The scaling of I_{syn} is done by multiple current mirrors. These are – in their simplest form – circuits of two transistors. The current flowing trough the reference transistor I_{ref} controls the current flowing to the output transistor I_{out} . This current mirror will scale the current by the ratios of areas A_{ref} and A_{out} of the transistor. The mirrored current is

$$I_{\text{out}} = I_{\text{ref}} \frac{A_{\text{out}}}{A_{\text{ref}}}. \tag{5.2}$$

Also multiple output transistors can be placed in parallel, each having an individual I_{out} giving its size A_{out}. The design also has the advantage that no current flows between the transistors. In a distributed current mirror, the transistors are placed apart to bring a current on a different side on the chip.

Scaling from V_{gmax} to I_{syn} is done by a series of such current mirrors, as shown in Figure 5.8. The first two mirrors scale the current from the floating gate cell by a factor of g_{scale}. These are located in the floating gate block and mirrored into the synapse drivers. The last current mirror is distributed between synapse driver and synapse. Both sides are built of multiple transistors of different sizes, where each of the transistors can be disabled by an individual switch. Depending on the number and size of the active transistors, this current mirror scales the current then by

$$\frac{w}{g_{\text{div}}}.$$

The sizes of the transistors are chosen such that g_{div} can scale from 1 to 30 and such that w can scale from 1 to 15. Placing the right side of the current mirror in the synapse ensures that the final current I_{syn} is there directly drawn from the supply

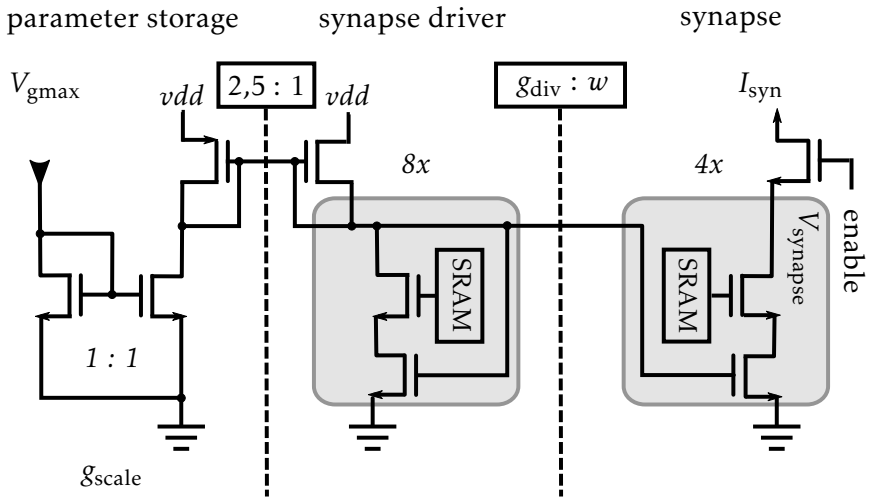


Figure 5.8 The scaling of the synaptic current I_{syn} involves three factors. The first factor is g_{scale} applied by the distributed current mirror between parameter storage and synapse driver. The second and third factors are w/g_{div} applied by the current mirror distributed between synapse driver and synapse. It can be scaled, as it is built from multiple transistors with different sizes in the ratio of: $1+1+2+2+4+4+8+8 : 1+2+4+8$. The scaling factor g_{div} is shared for the hole synapse row and V_{gmax} can be selected from 4 global parameters per quadrant of the chip.

voltage of the chip. It avoids that the synapses interfere with each other by depleting the current source. The actual current pulse is then triggered by a switch between synapse and neuron. If the synapse fires, this switch is active for the duration of t_{syn}.

5.3.2 Circuit Behavior

The current pulse generated by a synapse is supposed to be of rectangular shape, but the actual circuit shows a different behavior as an exemplary pulse shows in Figure 5.9. For this pulse we first notice the very high ascent of the pulse raising up to 60 000 nA at the beginning of the pulse, that last about 2 ns. This is followed by a plateau at about 250 nA for the remaining 8 ns. The pulse then closes with a very short spike, that is about 2000 nA high but so short, that it can be neglected. This current course is – with varying current values – similar for the whole parameter range of the synaptic input.

As we could verify using our simulations, the reason for the strong peak lies in the synapse. If the enable switch is off, the voltage in the synapse $V_{synapse}$ drops in less than t_{syn} down to 0 V, as it gets disconnected from its current source. If now the synapses fires, its parasitic capacity causes the strong current pulse as it charges up to the synaptic reference voltage V_{syn} again. This takes about 2 ns and afterwards

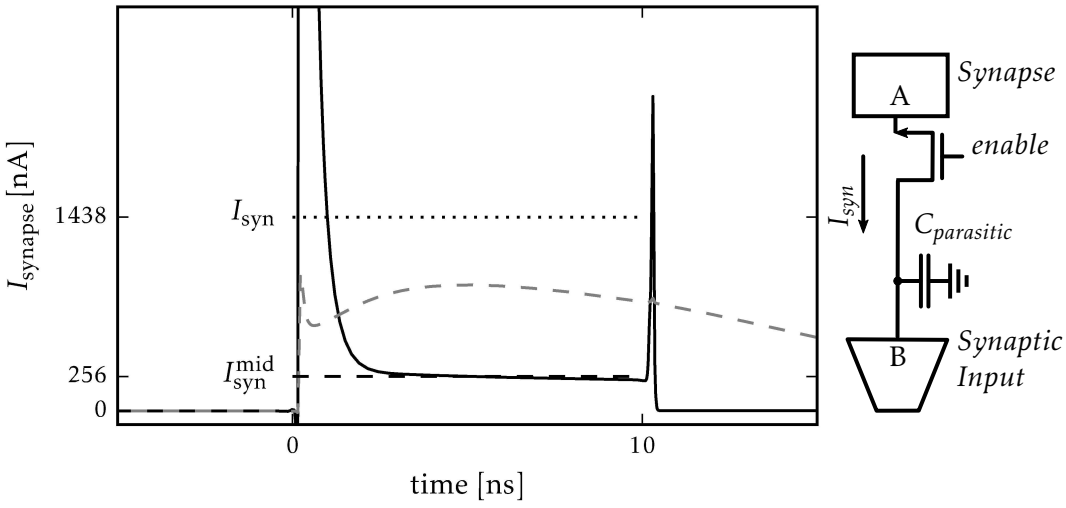


Figure 5.9 A simulation of the current pulse I_{synapse} as generated by the synaptic circuit (HICANN revision 4.1). Aside a simplified schematic of the synapse line, which connects the synapses to the neuron, is shown. The simulation contains a capacitor $C_{\text{parasitic}}$ to take the parasitic capacitance of the synapse line into account, that connect all synapse with the synaptic input. The current leaving the synapse (——) at A has a very large ascent (up to 60 000 nA) at the begin of the pulse, while the current flowing into the synapse (– –) at B is smoothed out. Note that the course of the currents in this simulation likely differs from the real currents in the chip, especially at the neuron side, because this is a transistor-level simulation with only a single synapse. But as the charge is preserved in the simulation, our final result is not impaired by this. Since the time scale of the pulse is 2 to 3 magnitudes smaller than the synaptic time-constants, the actual shape has no effect on the final PSP. We therefore use the average current I_{syn} to characterize the strength of the synaptic pulses.
The sharp ascent occurs because the synapse circuits behind the enable switch are drained to 0 V by leakage currents. When the synapses fires its circuits have to been charged to the potential of the synapse line V_{syn}. We take the current at the mid of the pulse $I_{\text{syn}}^{\text{mid}}$ as reference for the pulse strength without this charging effect.
The parameters were: $V_{\text{gmax}} = 1\,\mu\text{A}$, $w = 8$, $g_{\text{div}} = 11$ and $C_{\text{parasitic}} = 1\,\text{pF}$.

the intended current flows to the synapse. As we see later this is unfortunate because this peak now dominates the total current flowing into the synaptic current. As the length of this charging pulse does not depend on the PLL its effect becomes more dominant for faster PLL settings.

Further, we can observe that the shape of the current pulse, as it arrives at the synaptic input, becomes smoothed out into an about four times longer bulge. This is caused by the parasitic capacity of the synapse lines, which is about 1 pF. However, the total amount of charge doesn't change. Further the simulation here is limited to the transistor level, therefore the actual shape as it occurs in HICANN is different. Lasty, the length of the pulse is about 2-3 magnitudes shorter than the synaptic time-constant itself. Therefore the final shape of the PSP is not affected by actual shape of the current pulse. We take advantage of this and characterize its strength by its mean current I_{syn}, and discard the actual shape of pulses.

For a systematic comparison with the intended design we conducted simulations for the whole parameter space. We simulated for $V_{\text{gmax}} = 200$ to 2000 nA in steps of 200 nA , $g_{\text{div}} = 2$ to 30 in steps of 2 and all settings for the synaptic weight $w = 0$ to 15. This simulation is done for the revision 4.1 of HICANN. As described in Section 2.2.3, the g_{scale} is ten times larger in HICANN revision 2. Therefore the results presented here are also valid for revision 2, but only within a range for $V_{\text{gmax}} = 20$ to 200 nA.

Figure 5.10 shows the effect of the parameters for $V_{\text{gmax}} = 600$ nA. The observed I_{syn} shows a monotonic decay for larger g_{div}. However, for the parameter w, I_{syn} rises not monotonically and is not zero for weight $w = 0$. This is caused by the charging of the synapse at the begin of the pulse.

We can also confirm the non monotonically growing weights with a measurement on HICANN. As shown by Equation (2.8), the height h of the PSP is directly proportional to the weight of the synaptic event. Figure 5.11 shows the average height over all neurons on HICANN revision 4.1 stimulated with spikes of varying w and g_{div}. It qualitatively confirms the results obtained from the simulation and also shows that the resulting PSPs are actually quite homogeneous.

5.3.3 Analysis of the Charging Effect

A direct look at the results shows us that the direct comparison with Equation (2.18) is not a promising approach. We have to extend the model to take the initial ascent into account. First we separate the data into to parts, which we can analyze independently from each other. We assume that the plateau of the pulse has the current which we expect from Equation (2.18). Therefore we define $I_{\text{syn}}^{\text{mid}}$ to be the current at the middle of the pulse, as shown in Figure 5.9. Next we can define the current in the peak of the ascent as

$$I_{\text{syn}}^{\text{peak}} = I_{\text{syn}} - I_{\text{syn}}^{\text{mid}}. \tag{5.3}$$

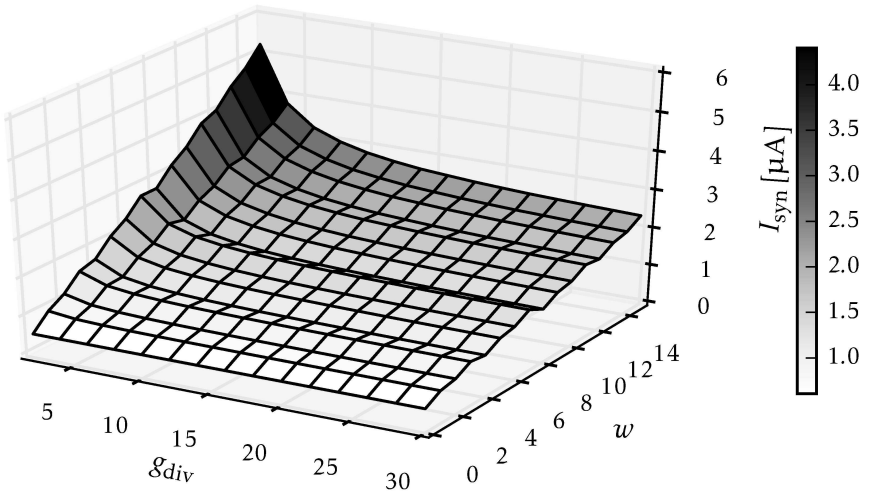


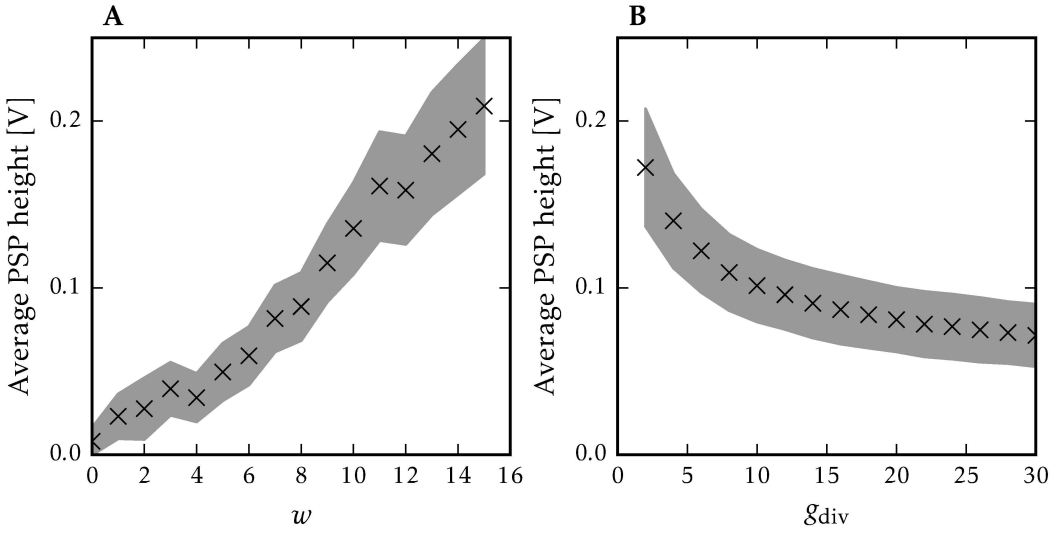
Figure 5.10 Effective synaptic current I_{syn} for varios settings of V_{gmax}, w and g_{div}. The non-monotonous ascent of the weights observed in the measurements is clearly visible as well as the offset. Otherwise we observe the expected behaviour, given by Equation (2.18).

In Figure 5.12 $I_{\text{syn}}^{\text{mid}}$ is compared to I_{syn} for selected parameter settings. It shows that the strength of $I_{\text{syn}}^{\text{peak}}$ mostly depends on w, which supports the assumption that the peak is caused by the charging the parasitic capacity of the synapse circuit.

We created a model to verify this assumption: Each bit of the digital weight value w adds a different transistor to the capacity. Therefore the total strength of the peak should be dependent on the active bits in the weight value. The resulting current should be linearly proportional to the total capacity. So we can assume that the peak current $I_{\text{syn}}^{\text{peak}}$ is composed as

$$I_{\text{syn}}^{\text{peak}} = i_0 + i_1 w_1 + i_2 w_2 + i_4 w_4 + i_8 w_8, \tag{5.4}$$

where $w_n = 1$, if n-th bit of w is set and otherwise $w_n = 0$, and i_n is the recharging current caused by the corresponding capacity, averaged over the length of the complete pulse t_{syn}. We fitted this model to the data obtained in the simulations and show the result for two different g_{div} settings in Figure 5.13. The model describes the data well because the standard deviation of residuals is only about 17 nA. This shows that the peak is indeed caused by charging effect of the synapse.



	V_{gmax}	w	g_{div}
A	0.06 µA	0 to 15	30
B	0.97 µA	3	2 to 30

Figure 5.11 Measurement of the effect of the synaptic weight w and the divisor g_{div} on 512 on High Input Count Analog Neuronal Network (HICANN). The plots show the averaged PSP height over all neurons on a single HICANN, the gray area marks the standard deviation. We observe unexpectedly, that the weights have non-monotonic steps. To compensate variations between the synapses, each neuron is stimulated using 3 synapses in parallel. Note that the offset for small weights can be dampened by a conservative calibration of V_{convoff} as discussed in Section 7.2. The used parameters are listed in Appendix A.5.7.

5.3.4 Extended Model for the Synaptic Strength

With the description of the charging effect, we can now compare the data set to the original model, if we add Equations (2.18) and (5.4). We obtain a synaptic current of

$$I_{\text{syn}}\left(V_{\text{gmax}}, g_{\text{div}}, w\right) = V_{\text{gmax}} \cdot g_{\text{scale}} \cdot \frac{w}{g_{\text{div}}} + i_0 + i_1 w_1 + i_2 w_2 + i_4 w_4 + i_8 w_8, \tag{5.5}$$

where we determine the parameters i_n and g_{scale} by fitting against the complete data set. We define the relative error of the fit for each data point as

$$\varepsilon_{\text{rel}} + \frac{I_{\text{syn}}^{\text{fit}} - I_{\text{syn}}}{I_{\text{syn}}}, \tag{5.6}$$

where $I_{\text{syn}}^{\text{fit}}$ is the current given by the fitted model.

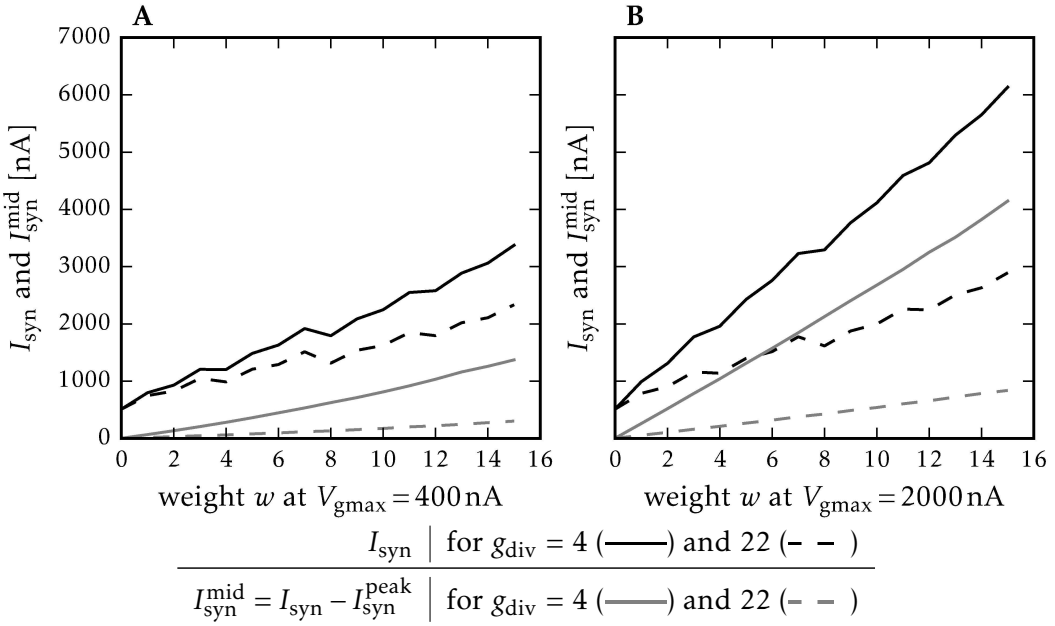


Figure 5.12 Simulation the total current I_{syn} (——) and the current $I_{syn}^{mid} = I_{syn} - I_{syn}^{peak}$ (——). We can well observe that the remaining current I_{syn}^{mid} rises almost linearly with the weight and that I_{syn}^{peak} acts as offset on that current.

In Figure 5.14 the obtained parameters are shown. We obtain a distribution of the relative errors of $\varepsilon_{rel} = 0.00 \pm 0.25$, but we find that the parameters i_n deviate more than 10 % from the ones obtained by the analysis of the peak.

Therefore we decide to extend the model by correction terms. Analyzing the residuals, we find an additional non linear dependency of the weight as major reason for the deviation between Equation (5.5) and the data, which we compensate by adding a linear and quadratic correction. Futher we find a deviation for g_{div}, which we can best compensate adding an exponent to g_{div}. Adding those to Equation (5.5), we obtain

$$\begin{aligned} I_{syn}\left(V_{gmax}, g_{div}, w\right) = & V_{gmax} \cdot g_{scale} \cdot \frac{w}{{g_{div}}^{\gamma}} + \frac{\beta_1 w + \beta_2 * w^2}{g_{div}} \\ & + i_0 + i_1 w_1 + i_2 w_2 + i_4 w_4 + i_8 w_8, \end{aligned} \tag{5.7}$$

where γ, β_1 and β_2 are further parameters to be determined by the fit.

Figure 5.14 shows also the result of the second fit. We could reduce the spread of the distribution of the relative errors to $\varepsilon_{rel} = 0.0 \pm 0.1$. This improvement justifies the three additional parameters. Further we observe, that the parameters describing the initial peak agree better with the ones shown in Figure 5.13. Lastly, we note that the strongest variations occur for large weights and small $V_{gmax} < 400\,nA$.

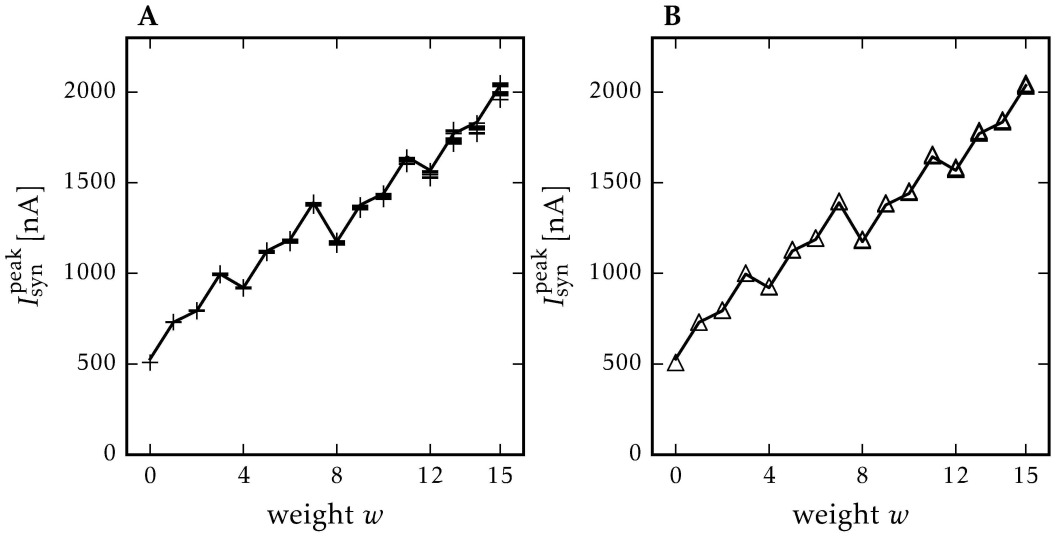
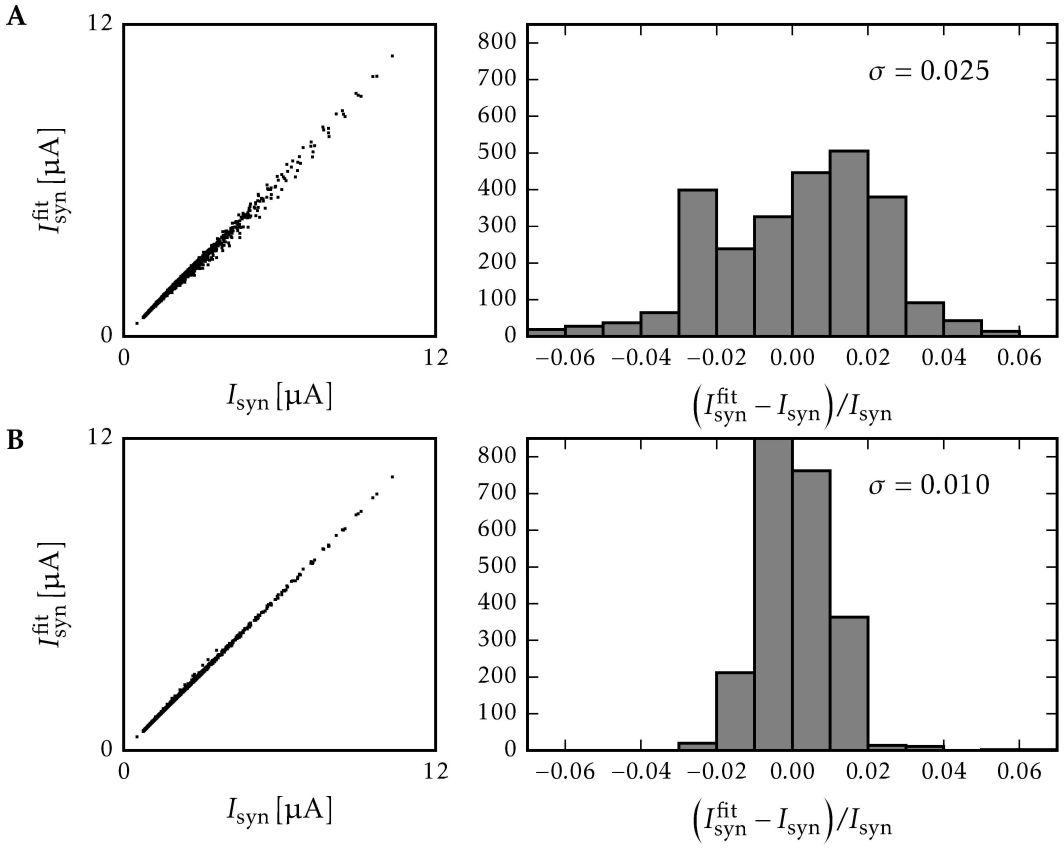


Figure 5.13 To obtain the parameters for Equation (5.4) we fit the equation against all $I_{\text{syn}}^{\text{peak}}$ values we obtained from simulation presented in Section 5.3.2. The plots each show a subset of the data, in **A** for $g_{\text{div}} = 4$ and in **B** for $g_{\text{div}} = 30$. Both are nearly identical, because the charging effect is not affected by g_{div} setting. The multiple points for each weight w are from different V_{gmax} settings. As the fit is done using the complete data set it is identical in both plots. The obtained parameters are: $i_0 = 527\,\text{nA}$, $i_1 = 203\,\text{nA}$, $i_2 = 266\,\text{nA}$, $i_4 = 394\,\text{nA}$ and $i_8 = 647\,\text{nA}$. The residuals, which are not shown, have a standard deviation of 16.9 nA.

5.3.5 Variations of the Synaptic Strength

Because we use spike input for the calibration of the synaptic inputs, the variation we can expect in the strength of I_{syn} caused by mismatch are of interest. To estimate these, we repeat the simulation described in Section 5.3.2 as Monte Carlo simulations using $N = 500$ samples for each parameter combination. However we restrict the parameters to $V_{\text{gmax}} = 600$, 1000 and 1400 nA because of the much longer simulation time needed. Further we exclude the scaling factor g_{scale} from the variations applied by the simulator because it is shared for all synapses stimulated by the synapse drivers in a quadrant of HICANN. We further assume that synapse and synapse driver variate independently, which is generally not the case on HICANN. Each of the two individually controllable g_{div} settings per synapse driver is used for one synapse row with 256 synapses. However, if we use neurons that are not interconnected, no synapses of these neurons share the g_{div} parameter with each other.

To evaluate the simulation we fit Equation (5.7) to each of the 500 Monte Carlo samples. We here fix $g_{\text{scale}} = 0.42$ as obtained by the typical simulations for two

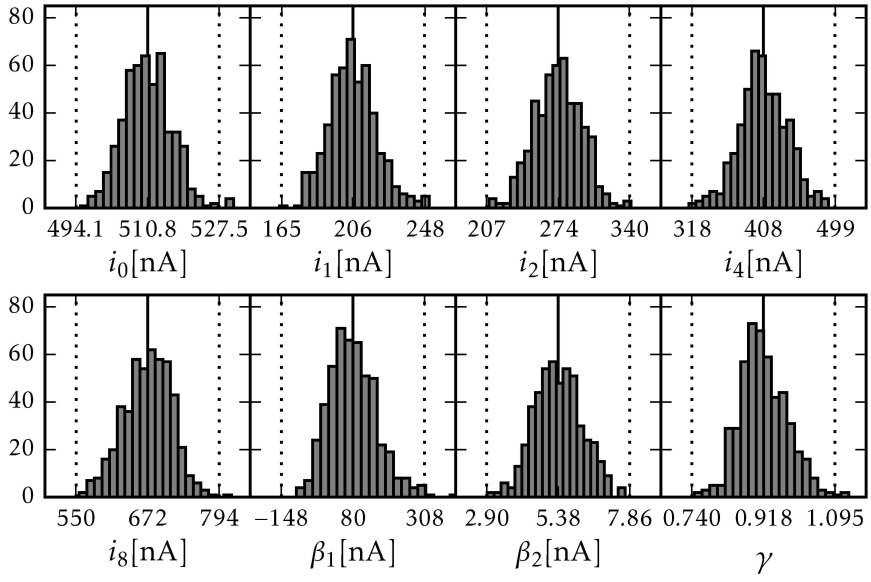


	i_0 [nA]	i_1 [nA]	i_2 [nA]	i_4 [nA]	i_8 [nA]	g_{scale}	β_1 [nA]	β_2 [nA]	γ
A (Eq. 5.5)	501	220	300	458	773	0.52			
B (Eq. 5.7)	520	208	276	410	678	0.42	73	4.93	0.92

Figure 5.14 Comparison of the results of the simulation with the two proposed models. **A** Data fitted to basic model given by Equation (5.5). 47 data points are outside of the histogram. **B** Data fitted to the extendend model using corrections for g_{div} and w as given by Equation (5.7). 8 data points are outside of the histogram. The table show the parameter obtained by the fit. The parameter are described in Section 5.3.2 and we fit to 2640 data points in total.

reasons: Firstly, it is excluded from the variations in the simulation and therefore should not vary between the samples. Secondly, strong correlation with the γ parameter occurred in the fit occurred because we used too few distinctive V_{gmax} values. The resulting distributions are shown in Figure 5.15. The obtained mean values agree, with small deviation, with the results from the typical case. The spread of most distributions is small, except for the correction factor β_1.

For practical use we want to know how strong the resulting currents variate.



Parameters of N=500 fits:

i_0	i_1	i_2	i_4
510.8 ± 5.6 nA	206 ± 14 nA	274 ± 22 nA	408 ± 30 nA
i_8	β_1	β_2	γ
672 ± 41 nA	80 ± 76 nA	5.38 ± 0.83 nA	0.918 ± 0.059

Figure 5.15 Monte Carlo simulation for the parameters show in Section 5.3.5. N = 500 samples were taken and to each of it Equation (5.7) was fitted. The scaling factor g_{scale} is excluded from the variations, because it is shared between multiple synapse drivers. Therefor is $g_{\text{scale}} = 0.42$. The histogram show the obtained redistribution for each parameter. The mean and 3-sigma are annotated. The table list the mean and the standard deviation for each parameter. The obtained results agree well with the typical simulation.

Because we have a 3-dimensional parameter space, we define $(\sigma/\mu)_I$ as the standard deviation of all samples of $I_{\text{syn}}\left(V_{\text{gmax}}, g_{\text{div}}, w\right)$ normalized by its mean. The resulting distribution is shown in Figure 5.16. For most parameters of the simulated parameter sets, $(\sigma/\mu)_I$ is well below 0.04, which means that for about 70 % of the simulated parameter sets all values lay within 10 % of the target values. The largest $(\sigma/\mu)_I$ is 0.10, which corresponds to a maximal deviation of about 30 %.

Further, we find a strong dependency of the spread on g_{div}, which is also shown in Figure 5.16. In contrast, the other parameters do not show correlations with $(\sigma/\mu)_I$. This is possibly caused by the size dependency of the transistor mismatch. For a divisor of $g_{\text{div}} = 2$ only the two smallest transistors are active in the current mirror. These are subject to larger variations than the transistors with larger area [Pelgrom et al. 1989]. For large g_{div} the total area of the involved transistors rises, reducing

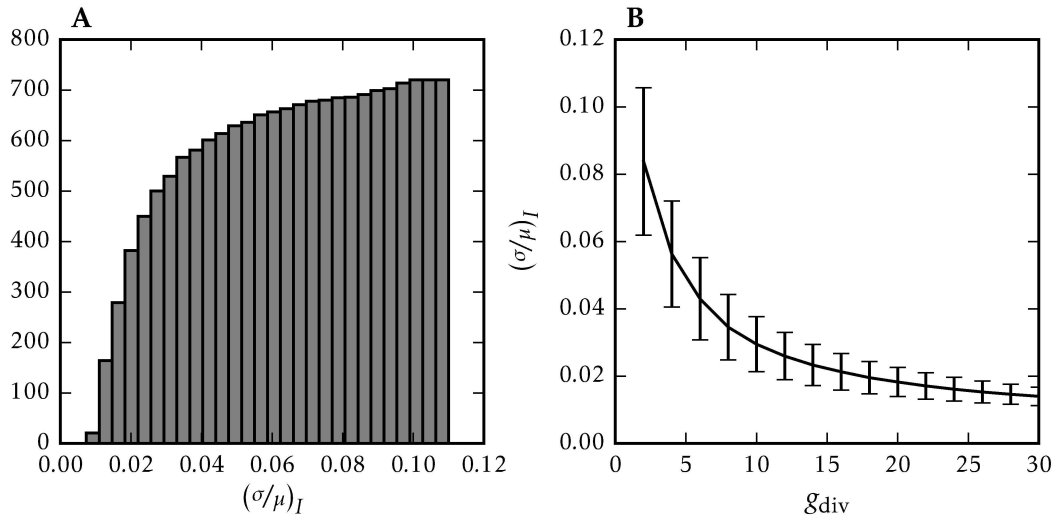


Figure 5.16 Relative deviation from the mean synaptic current I_{syn} obtained from a Monte Carlo using 500 sampels. For each parameter set we normalized standart deviation with the mean synatic current and obtain $(\sigma/\mu)_I$. **A** Cumulative histogram of the resuting $(\sigma/\mu)_I$ values. **B** Mean $(\sigma/\mu)_I$ with standart deviation for different g_{div} settings. We note, that smaller g_{div} lead to an larger deviation. For other the other parameters V_{gmax} and w we don't observe such a correlation.

the total susceptibility to mismatch.

5.3.6 Summary

We find that the synapse shows an offset caused by recharging the circuits in the synapse at the begin of each pulse. These manifest in a sharp peak at the begin of each current pulse. It is contributed to the small size of the synapses and the design using in the following generation HICANN - Digital Learning System (HICANN-DLS) does not show such a behaviour [Friedmann et al. 2016]. However the resulting non-monotonic dependency on the synaptic weight must be taken into account for learning rules, as those usually expect monotonic weights. An alternative approach can be to forbid certain weights, so that the remaining bit form monotonic relation.

Including this effect, we could create a well suited model for the occurring effects. An empirical correction factor allows us to predict the generated currents well. Monte Carlo simulations allows us to estimate the expected variation of the synaptic current I_{syn}, which is mostly below 10 % and gets smaller for large g_{div} divisors setting. So here the smallest possible settings should be chosen, whenever possible. We therefore chose this value also for the calibration of the synaptic time-constants. Further, the small variation of the synaptic strength is a good premise for this calibration.

With the synaptic input in revision 4 we can reach currents of over 12 µA. We cannot give the absolute value, as we did not completely max out the reference current V_{gmax}. The minimum current is given by the parameter i_0 of Equation (5.7) and is about 500 µA. Because V_{gmax} is an analog parameter, the target current can be chosen arbitrarily in between these ranges.

6 Synaptic Input on HICANN Revision 2

In this chapter we present our results of the characterization of the synaptic input and the synapses of the second revision of HICANN. Our goal is to extend the calibrations developed by Schwartz [2013] with calibration for the synaptic time-constants. Schwartz disabled both synaptic inputs completely in his calibrations. By enabling them we measured that they are not functioning correctly due to errors in its function introduced by mismatch. In Section 6.1 we will characterize the effect of mismatch on the synaptic input. Following in Section 6.2 we analyze the effects of synaptic events on the neurons. We were able to measure the synaptic time-constants, but conducted that the parameters cannot be calibrated independently from other parameters. Therefore we created in close cooperation with Mitja Kleider, Dominik Schmidt and Sebastian Schmitt a functional calibration, that tries to make as many neurons functional as possible. These calibration efforts are presented in Section 6.3. Finally, we discuss the limitations of the second revision of HICANN in Section 6.3.6.

The used HICANN parameters and settings are described in Section 4.2 and Appendix A.1. In addition, we resort to simulations of the analog circuits as described in Section 4.4.

6.1 Synaptic Input

The synaptic input circuit of HICANN was designed by [Millner 2012] to integrate synaptic events and generate exponentially conductance based PSPs. A description of the synaptic input circuit of HICANN is in Section 2.2.1 and its schematic can be seen in Figure 2.6. In previous work the synaptic input was only analyzed using transistor level circuit simulation [Millner 2012; Kiene 2014; Schwartz 2013]. We were the first to conduct systematical measurements of the circuit in real hardware.

In general, the synaptic input can produce well defined PSPs as defined by Equation (2.13), but the performance of these is strongly limited by the following effects: Firstly, the missing DC-offset correction of OTA_1 can cause either a permanent current to the membrane or weakens the PSP. Secondly, the synaptic input can only integrate a very limited number of synaptic events. Lastly, the maximal time-constant of the integrator circuit is limit, because its internal voltage will drastically rise for larger R_{syntc}. Therefore we will first present our result of measurements and simulations and afterwards discuss these in the conclusion in Section 6.1.4. A part of the simulations presented in the chapter were orignially developed by Kiene [2014] in the course of his bachelor thesis, but for this work we repeated and extended these simulations on our own.

6.1.1 Missing Offset Compensation in the Operational Transconductance Amplifier

One of our first observations while working on HICANN was that for about half of the neurons the resting potential is pulled towards one of the tow reversal potentials – even without any stimulus. This is caused by mismatch in the synaptic input circuits affecting the comparison between the reference voltage V_{syn} and integrator voltage $V_{\text{integrator}}$. One effect of mismatch is that the zero point of each OTA on VLSI device varies [Razavi 2001]. Additionally, the OP in the synaptic input circuit also introduces an offset between V_{syn} and the terminal V_+ of OTA_1. We take both effects into account by introducing in Equation (2.17) a for each instance of the synaptic input specific offset α_i. We obtain for the output current of OTA_1

$$I_{OTA_1} \propto g_{i/x} \propto V_{\text{syn}} - V_{\text{integrator}} + \alpha_i \quad \text{for} \quad I_{OTA1} > 0. \tag{6.1}$$

The actual value of α_i causes the functionality of the synaptic input to differ from the desired behavior. For inputs where $\alpha_i > 0$ synaptic events have to push $V_{\text{integrator}}$ over $V_{\text{syn}} + \alpha_i$. This weakens all synaptic events and causes vanishing small events. On the other hand for inputs where $\alpha_i > 0$ the bias current is always $I_{OTA1} > 0$. As consequence the synaptic conductance $g_{i/x}$ of the synaptic input is not 0, but has a minimum conductance $g_{i/x}^{min}$ value, which is always $g_{i/x}^{min} > 0$. This causes a permanent leakage current onto the membrane. It effectively behaves like the neurons see a constant pseudo activity from this input and fully explains membrane is pulled towards one of the reversal potentials.

Those effects can only be quantified using circuit simulations, as they were part of bachelor thesis of Kiene [2014] and well agree with our own simulations. The mismatch leads to a distribution of

$$\alpha_i = -14 \pm 24\,\text{mV},$$

which is shown in Figure 6.1.

Further of interest and also directly related to α_i is the minimum value for I_{OTA_1}. For only about one third of the neurons this is zero, for the others it goes up to over 400 nA. Because the OTA_0 of the synaptic input is the same OTA as for the leakage conductance, we can compare these: The medium range of possible reachable values for τ_m are set using bias currents for the leakage conductance I_{gl} from 100 to 400 nA. The simulations show, that the offset of OTA can affect neurons with a similar strength like the leakage conductance. Therefore a strong leakage conductance settings are needed to compensate this.

We can also observe this indirectly on the neuron membrane. The effect is visible in the following experiment, which also initially brought it to our attention. For this we use the possibility to disable the synaptic input by turning the bias for OTA_1 I_{conv} to 0 A. This allows us to test the influence of both inputs on the neuron. The measurement consists of four sweeps of the leakage conductance bias I_{gl} from 0 to 2 µA, one with both synaptic inputs enabled, one with only the excitatory, one

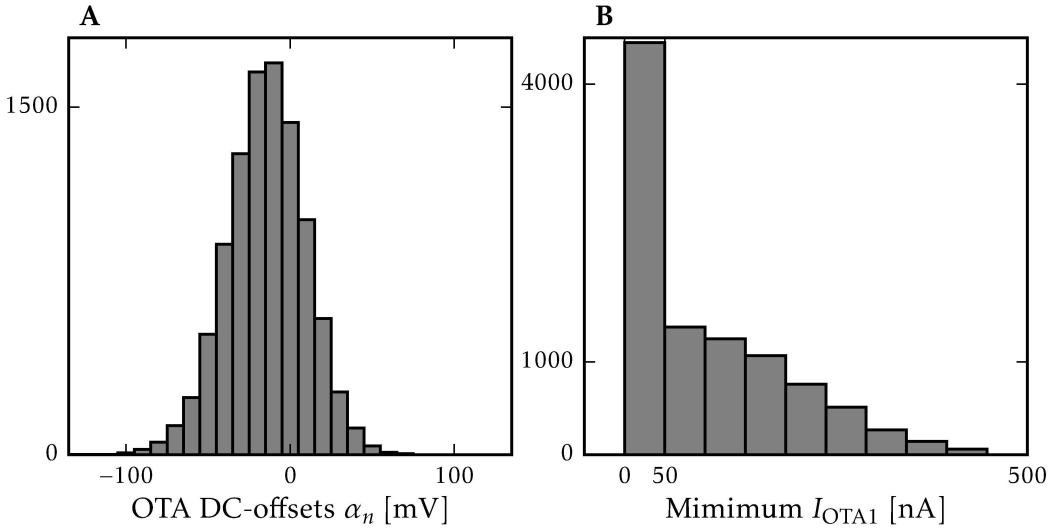


Figure 6.1 Monte Carlo simulation of the combined effect of the missing offset correction for the OP and for OTA_1 in the synaptic input. **A** Mismatch causes the circuit to have an DC-offset, while comparing the integrator voltage $V_{\mathrm{integrator}}$ with the reference voltage V_{syn}. The mean value and standard deviation of the offset are $-14 \pm 23\,\mathrm{mV}$. **B** For about 2/3 of the synaptic inputs, this leads to a minimum bias current I_{OTA_1} above zero for OTA_0. The bias current is proportional to the conductance of that input, as shown in Figure 2.4. Therefore, causes a constant leak current onto the membrane. Both simulations were conducted for $V_{\mathrm{syn}} = 1.0\,\mathrm{V}$.

with only the inhibitory and one with both disabled. The neurons have a resting potential $E_{\mathrm{l}} = 0.8\,\mathrm{V}$ and the reversal potentials $E_{\mathrm{syni}} = 0.6\,\mathrm{V}$ and $E_{\mathrm{synx}} = 1.0\,\mathrm{V}$, all with calibrations applied. The spiking threshold is set to $V_{\mathrm{t}} = 1.2\,\mathrm{V}$, higher than E_{synx} to avoid spiking. All other parameters have default values as described in Section 4.2.

Figure 6.2 shows the results of these measurements and compares them to a simulation of a simplified neuron consistent only of the synaptic inputs and the leakage OTA as presented by Kiene [2014]. The measurement shows, agreeing with the given simulations, that the minimum conductances g_i^{min} and g_x^{min} of most synaptic inputs are strong enough to disturb the resting potential. Equation (2.7) describes the effective resting potential E_l^{eff} as average of the three potentials weighted by the conductances. This shows that even large g_{l} cannot completely negate the effect of finite g_i^{min} and g_x^{min}, especially if we hold in mind that a part of the g^{min} reaches intensities in the medium range of g_{l} in the circuit.

Further it shows that, even if $g_i^{min} \approx g_x^{min}$ the effect on the membrane doesn't cancel out, both are in the denominator and have an impact on g_{l}. This also effects

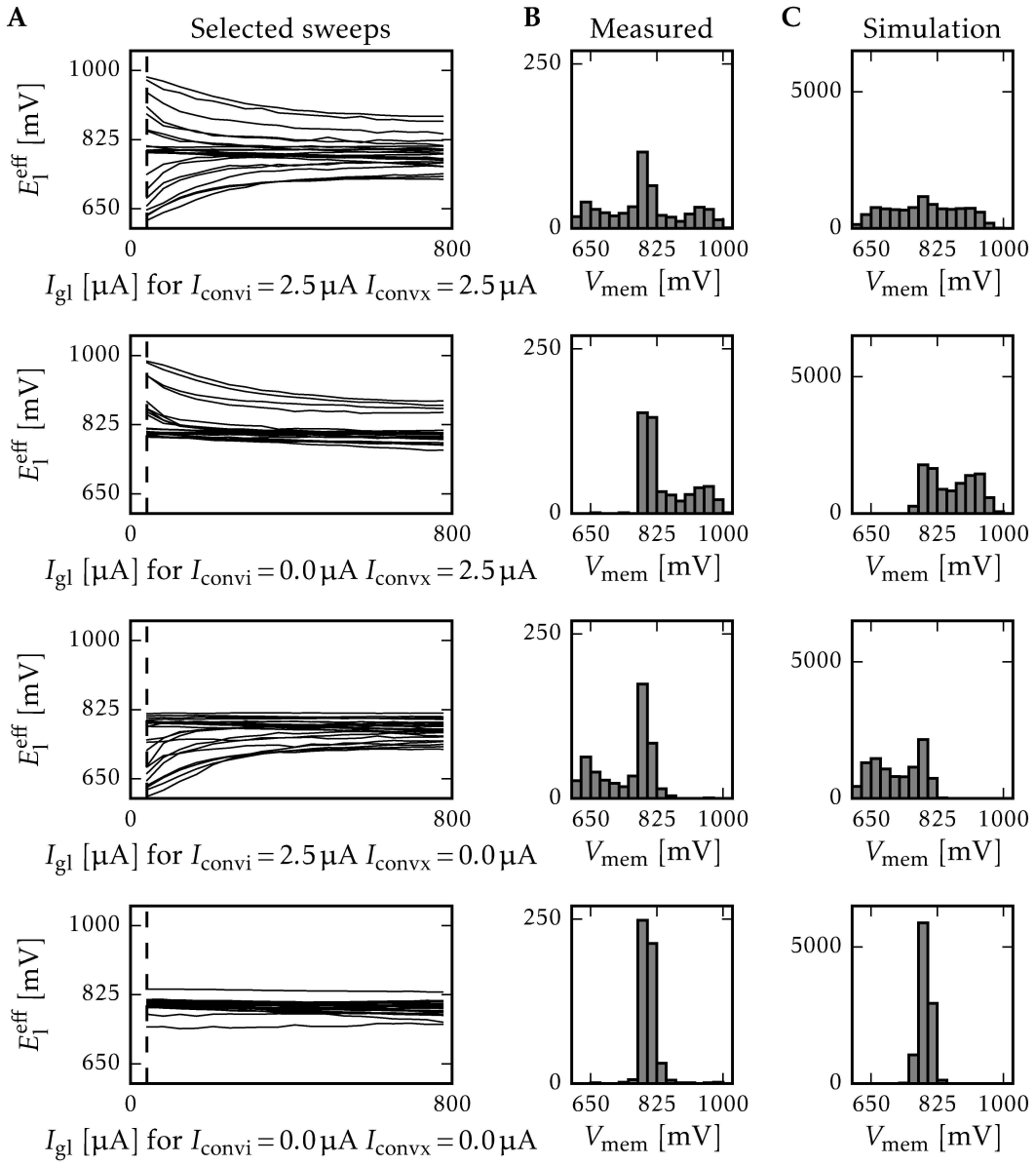


Figure 6.2 Effect of sweeping I_{gl} on the membrane potential for all four possible settings of enabled and disabled synaptic inputs: I_{gl} is sweeped from 0.1 to 1.8 µA in steps of 0.1 µA. For each step only I_{gl} was updated. Resting and reversal potentials are set to $E_l = 0.8\,V$, $E_{syni} = 0.6\,V$ and $E_{synx} = 1.0\,V$ with applied calibration. **A** Effect of sweeping I_{gl} on the effective resting potential of the neuron. Although all 512 neurons on a HICANN where measured, for clarity only every 23th neuron is show. For many neurons one of the reversal potential dominates over the leakage for smaller I_{gl} values. Disabling the respective synaptic inputs proves that they are the source of the distortion for the resting potential. **B** The histogram over all neurons at $I_{gl} = 0.1\,nA$ shows the quantity of neurons affected. **C** For comparison we repeated the simplified neuron simulation shown in [Kiene 2014] with the same parameters our measurement had. The distributions show a good agreement.

the effective membrane time-constant τ_m of the neuron. For a LIF neuron without external stimulus is

$$\tau_m = \frac{C}{g_i^{min} + g_x^{min} + g_l}. \tag{6.2}$$

As the denominator grows with non zero synaptic conductances, the membrane becomes faster than intended.

6.1.2 Integration of Synaptic Events

Next we want to analyze the handling and integration of the synaptic events by the synaptic input. Here we have to rely on simulations as the circuit is not accessible for direct measurements in HICANN. Previous work on this are the dissertation of Millner [2012] who designed this part of HICANN and the bachelor thesis of Kiene [2014]. Both only showed exemplary simulations, which also only partly match with the predicted behavior. To close this gap we conducted systematic simulations.

We begin with the analysis of the effect of a single synaptic event. For each a short current pulse I_{syn} of lengths of t_{syn} is generated in the synapse, details were already presented in Section 5.3 and for this HICANN revision I_{syn} can be set as large as 120 µA. The incoming current pulses I_{syn} are integrated by an operational amplifier integrator, which is part of Section 5.3. Then the voltage in the integrator $V_{\text{integrator}}$ is transformed by OTA_1 and OTA_0 to the synaptic current flowing onto the membrane.

The integrator consists of an OP, a capacitor C and the resistance R_{syntc}. The voltage in the integrator $V_{\text{integrator}}$ represents the integrated events. Technically, it is limited to 1.8 V. For an ideal circuit, where R_{syntc} is independent of the potential, this can be solved analytically: For a single pulse at time $t = 0$ the integrator voltage is

$$V_{\text{integrator}} = \begin{cases} V_{\text{syn}} - I_{\text{syn}} \cdot R_{\text{syntc}} \cdot \left(1 - \exp\left(\frac{t}{CR_{\text{syntc}}}\right)\right) & \text{for } t \leq t_{syn}, \\ V_{\text{syn}} - I_{\text{syn}} \cdot R_{\text{syntc}} \cdot \left(1 - \exp\left(\frac{t_{syn}}{CR_{\text{syntc}}}\right)\right) \cdot \exp\left(\frac{t_{syn}-t}{CR_{\text{syntc}}}\right) & \text{for } t > t_{syn}, \end{cases} \tag{6.3}$$

where V_{syn} is the reference voltage of the integrator [Millner 2012; Kiene 2014]. The time-constant in this circuit is given by $\tau_{\text{syn}} = CR_{\text{syntc}}$. We can use Taylor expansion to obtain the simplified equation

$$V_{\text{integrator}} = \begin{cases} V_{\text{syn}} - \frac{I_{\text{syn}} \cdot t_{syn}}{C} & \text{for } t \leq t_{syn}, \\ V_{\text{syn}} - \frac{I_{\text{syn}} \cdot t_{syn}}{C} \cdot \exp\left(\frac{-t}{CR_{\text{syntc}}}\right) & \text{for } t > t_{syn}, \end{cases} \tag{6.4}$$

as long as $t_{syn} << \tau_{\text{syn}}$ [Millner 2012; Kiene 2014].

The resistor R_{syntc} is variable, which allows us to set the synaptic time-constant t_{syn}. It is controlled by the voltage parameter V_{syntc}. Its resistance grows exponen-

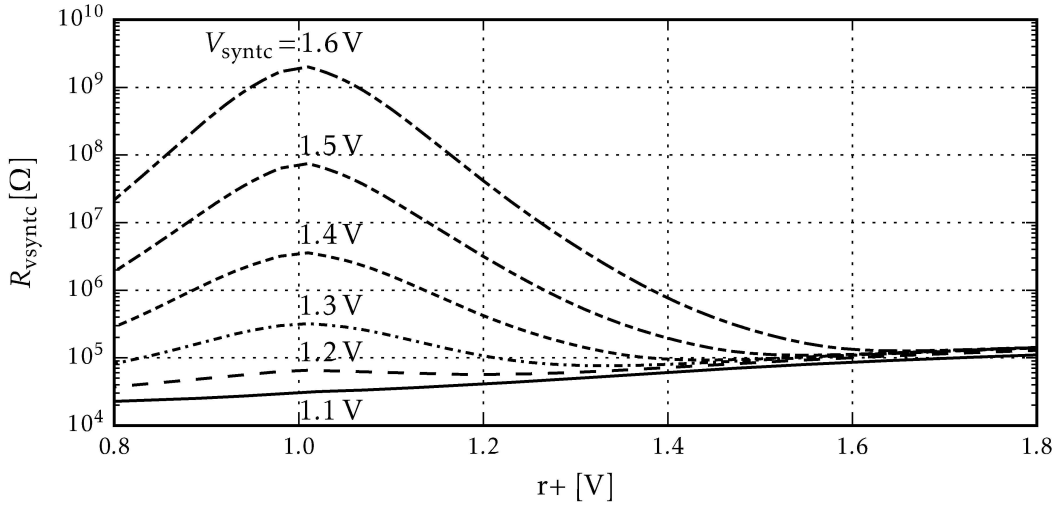


Figure 6.3 Simulation of the resistive element in the synaptic input for six different values of V_{syntc} from 1.1 to 1.6 V. The terminal r+ is sweeped, while the other terminal is held at r- = 1.0 V. R_{vsyntc} rises exponentially with the controlling voltage V_{syntc}. However, it is not stable against voltage changes in the integrator, effectivly limiting the maximum voltage and reducing the time-constant of the integrator.

tially from about $3 \times 10^4\,\Omega$ at $V_{\text{syntc}} = 1.1\,\text{V}$ to about $2 \times 10^9\,\Omega$ at $V_{\text{syntc}} = 1.6\,\text{V}$. Its characteristics are shown in Figure 6.3. The functional range is limited to a range of about 1.2 to 1.5 V, that would correspond to τ_{syn} from 0.02 to 18 µs. For lower voltages R_{syntc} is so small, that any current pulses decay almost instantaneously. The effect of larger R_{syntc} is shown in Section 6.1.3. The small range of 0.3 V is highly problematic as R_{syntc} varies four orders of magnitudes. For a try-to-try precision of x in floating gate cell the time-constant will vary about y. Further we have to reevaluate the validity of Equation (6.4). For $\tau_{\text{syn}} >> 10\,\text{ns}$ this requires $R_{\text{syntc}} \gg 4 \times 10^5\,\Omega$, which is reached at $V_{\text{syntc}} \approx 1.32\,\text{V}$. As we see later, the approximation of Millner [2012] won't hold for $V_{\text{syntc}} < 1.45\,\text{V}$.

Furthermore, we found out that R_{syntc} also depends on voltage across the resistor. As shown in Figure 6.3, it declines the larger the voltage becomes. For a difference of 0.2 V, which is still a reasonable input for OTA_1, it can fall over a magnitude depended on V_{syntc}. We expect that this effect is the most prominent reason for deviations from the behavior predicted by Equation (6.3), especially for larger I_{syn}.

In Figure 6.4 we show exemplary traces of $V_{\text{integrator}}$. The traces illustrate both effects, we can well observe the strong variation in possible time-constants. For a current pulse of $I_{syn} = 4\,\mu\text{A}$ and $V_{\text{syntc}} > 1.5\,\text{V}$ the rise of $V_{\text{integrator}}$ is already strong enough to lead to non-exponential decay due to the voltage dependency of R_{syntc}. Further we see a clear dependency on V_{syntc} for the rise of $V_{\text{integrator}}$. The same hold

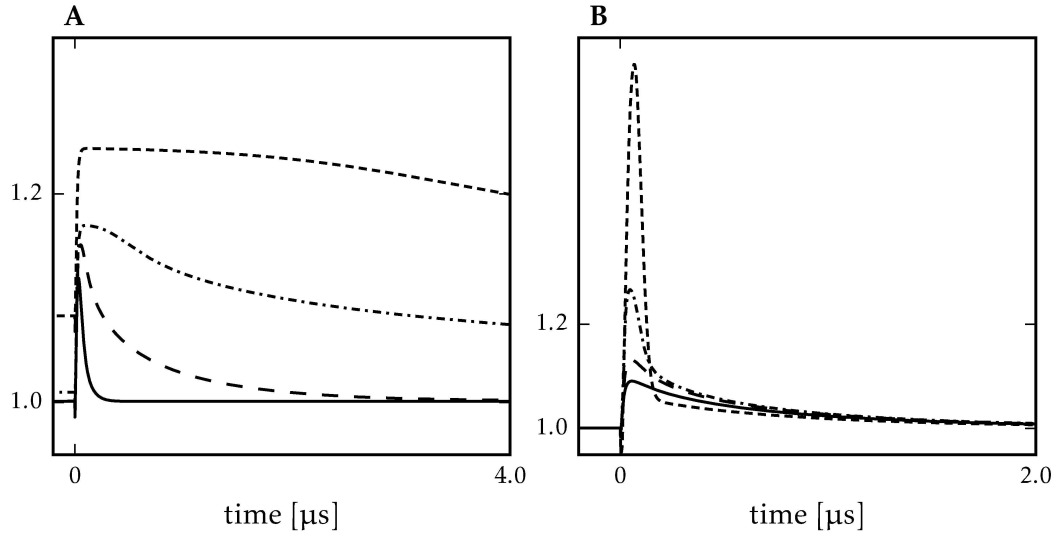


Figure 6.4 Simulations of response in the integrator for synaptic current pulses of $t_{syn} = 10$ ns. The simulation takes the parasitic capacity between synapse and synaptic input into account, which distorts the pulses, see Figure 5.9. **A** We show the large possible variation in time-constants for a given pulse strength. **B** We show the effect of pulses of of different strengths. Pulses with a large strength can saturate the synaptic input immediately. Because R_{vsyntc} becomes much smaller for such such large integrator voltages $V_{\text{integrator}}$ the pulse decays then immediately.

for the two current pulses with $I_{syn} = 3.5$ and 7.3 µA and $V_{\text{syntc}} > 1.4$ V. The stronger one will cause a higher rise of $V_{\text{integrator}}$, but will then fall fast to the same level as the smaller one having the same long term behavior. We also observe, that we can saturate the integrator easily with a single current pulse of $I_{\text{syn}} = 22$ µA which is, as shown in Section 5.3, well in the medium range of the I_{syn}. This pulse does not just causes the integrator to saturate, but also it effectively decays faster than the smaller pulses. This is caused by a voltage drop on the synapse site due to the large pulse, which increases the voltage difference across the integrator even further.

We conducted further simulations to clarify the dependency of the rise of $V_{\text{integrator}}$ on V_{syntc} and I_{syn}. For the ideal circuit this is equivalent to $V_{\text{integrator}}(t_{syn})$. But because we have a finite integration time in the simulation, we define the rise of $V_{\text{integrator}}$ as

$$\Delta V_{\text{integrator}} = max(V) - V_{\text{syn}}. \tag{6.5}$$

In the actual circuit the maximum is reached a bit later, due to finite gain of the

OP. As shown in Figure 6.4, for $V_{\text{syntc}} < 1.45\,\text{V}$ the strength of the synaptic pulse $\Delta V_{\text{integrator}}$ decreases continuously and is at only about half as strong $V_{\text{syntc}} = 1.2\,\text{V}$. In this range the simulated $\Delta V_{\text{integrator}}$ is smaller than the predicted values and the reaches it for $V_{\text{syntc}} > 1.45\,\text{V}$. The same holds for all tested value of I_{syn}.

An early saturation effect is noticeable too: For the smaller $V_{\text{syntc}} = 1.2$ and $1.3\,\text{V}$ the rise $\Delta V_{\text{integrator}}$ saturates at 0.25 and 0.34 mV. This is again caused by the dependency of R_{vsyntc} on the voltage difference across.

So far we considered only single spikes, but the synaptic input is actually intended to be used with large quantities of spikes. The maximum integrator voltage $V^{\text{max}}_{\text{integrator}}$ can be found to be

$$V^{\text{max}}_{\text{integrator}} = f_{\text{stim}} \cdot R_{\text{syntc}} \cdot I_{\text{syn}} \cdot t_{\text{syn}}. \tag{6.6}$$

It describes the balance between the decay of $V_{\text{integrator}}$ and the stimulus rate [Kiene 2014]. This holds for small excitation and is still approximately valid for larger ones.

To study the effect we simulated with a very fast input, the maximal rate a single synapse driver may achieve, $f_{stim} = 50\,\text{MHz}$. The effect on the integrator is shown in Figure 6.5 for two exemplary settings. In comparison, $V^{\text{max}}_{\text{integrator}}$ is about twice as large as $\Delta V_{\text{integrator}}$ for a single spike. We also observe for $V_{\text{syntc}} > 1.45\,\text{V}$ an overshoot effect, which is likely caused by the frequency response of the OP. It causes the equilibrium to occur only after up to 0.4 µs of spikes. We can also observe this in simulations with much lower input rates. The effect is strong enough to be visible on the membrane, but might be irrelevant for more realistic spike input.

We simulated this for a wide range of I_{gl} and V_{syntc} settings. To compensate the charging effects, we let the integrator settle to equilibrium before determining $V^{\text{max}}_{\text{integrator}}$, as indicated in Figure 6.5. The results are shown in Figure 6.6.

For $I_{\text{syn}} < 7\,\mu\text{A}$ we can observe an almost linear relation between $V^{\text{max}}_{\text{integrator}}$ and V_{syntc}, as we would expect from Equation (6.6). For $I_{\text{syn}} > 7\,\mu\text{A}$ the integrator voltage $V^{\text{max}}_{\text{integrator}}$ stagnates and then begins to fall again. The voltage dependency of R_{syntc} is the cause of this effect and it can partially compensated with larger I_{syn}. Further, $V^{\text{max}}_{\text{integrator}}$ quickly saturates for V_{syntc} settings larger than about 1.5 V. Conclusive we can say, that $V^{\text{max}}_{\text{integrator}}$ follows the predictions given by Equation (6.6) as long as $I_{\text{syn}} < 7\,\mu\text{A}$ and $V_{\text{syntc}} < 1.5\,\text{V}$.

6.1.3 Limit for the Synaptic Time-Constant

The strength of the resistor R_{syntc} sets a limit to the maximum time-constant the synaptic input can achieve. As shown in Figure 6.8, the internal integrator voltage $V_{\text{integrator}}$ begins to rise for $V_{\text{syntc}} \gtrsim 1.5\,\text{V}$. This increases the minimal conductance of the synaptic input leading do similar effects like the DC-offset of OTA_1. Therefore the actual usable range of V_{syntc} is limited. We indirectly observe this effect in measurements, as the resting level of the membrane gets shifted upwards at larger V_{syntc} settings. This can be seen for example in Figures 4.3 and 6.12.

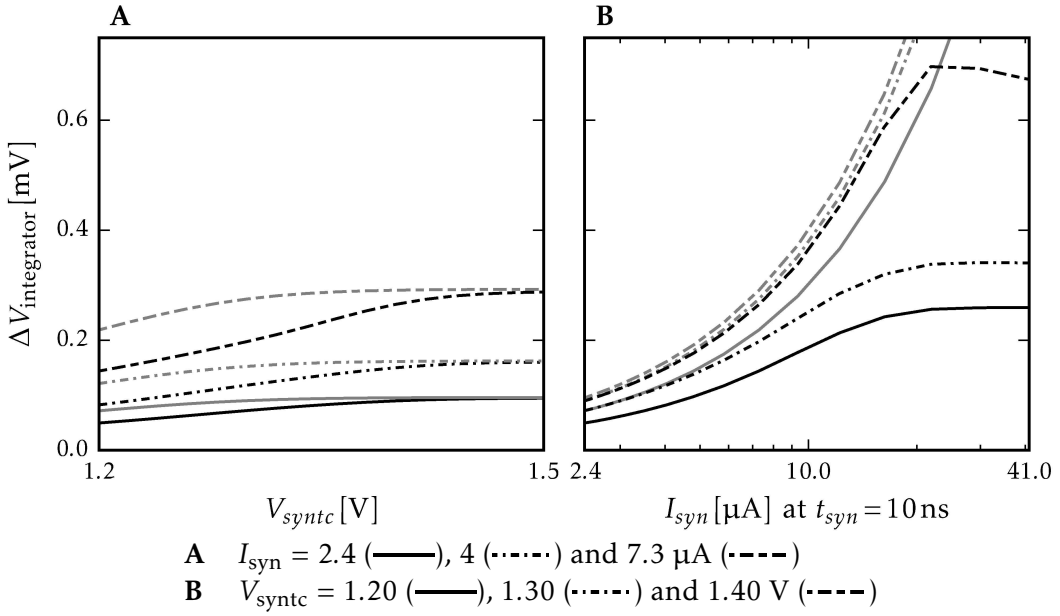


Figure 6.5 The plot shows strong synaptic pulses increase the integrator voltage $\Delta V_{\text{integrator}}$ depended on the setting of V_{syntc} and I_{syn}. Two exemplary traces of the dataset are shown in Figure 6.4. The gray lines represent the values predicted by Equation (6.3). For smaller resistances the pulse already decays significantly during the integration phase. This effect already sets on for larger V_{syntc} than expected, because the integrator needs about 50 ns to integrate the pulse. Also we observe that for pulse larger than about $I_{\text{syn}} \gtrapprox 10\,\text{us}$ $\Delta V_{\text{integrator}}$ no longer increase. This is caused by the voltage dependency of R_{syntc}.

The simulation indicates that permanently a very weak leakage current of several pA flows from synapse into the integrator. If the resistance R_{syntc} then becomes too large, it begins to block the flow of the current. If the circuit would had the OP this would simply cause a drop of the voltage in the synapse line. However the negative feedback loop of the OP will compensate the voltage drop by increasing the integrator voltage $V_{\text{integrator}}$. The Monte Carlo simulation shown in Figure 6.8 indicates that the threshold for a 2 mV rise of $V_{\text{integrator}}$ lays at $V_{\text{syntc}} = 1.489 \pm 0.014\,\text{mV}$. This matches well with the specified upper limit of $V_{\text{syntc}} < 1.45\,\text{V}$ [Millner 2012], even if this is effect is not explicitly describe there. In direct observations we observe many neurons that won't show a voltage shift for $V_{\text{syntc}} > 1.5\,\text{V}$, which is caused by a negative offset α_i of OTA_1. This also forbids direct comparison between simulation and measurements.

6.1.4 Summary

In summary we can say that the uncompensated DC-offset of OTA_1 has severe effects on the neuron dynamics. Firstly, for OTAs with a negative offset weak input will be

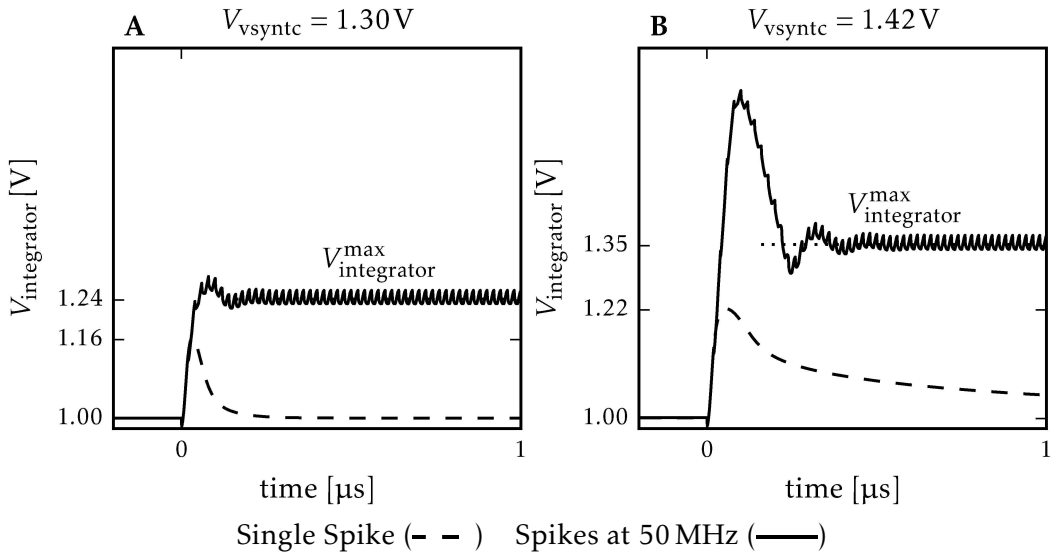


Figure 6.6 Simulations of response in the integrator for synaptic current pulses of $t_{syn} = 10\,\text{ns}$ with $I_{\text{syn}} = 5.83\,\mu\text{A}$. The synaptic input is stimulated with 50 MHz, which is the maximum possible rate. For comparison a single spike kis shown. The integrator needs to settle for higher settings of V_{syntc} This effect increase for $V_{\text{syntc}} > 1.45$ continuously. Therefor we define the maximum voltage of the integrator $V^{\text{max}}_{\text{integrator}}$ as the mean voltage after a short settling period of 0.5 µs.

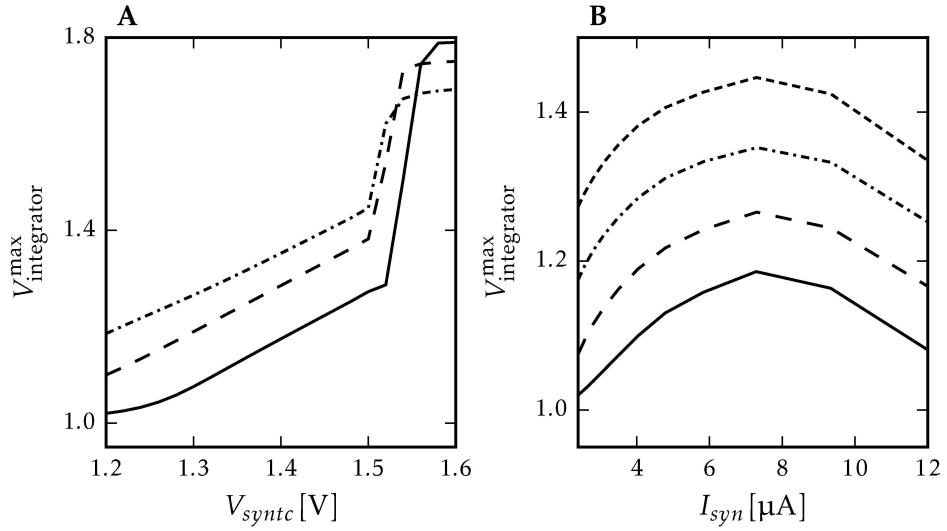
discarded. For a positive offset the situation is worse, as a high leakage conductance is required to compensate the permanent current from the input. However, we want to emphasize that Monte Carlo simulations were not yet available during the design phase of this HICANN. Therefore, the full impact of the DC-offset was unforeseeable at that time.

Secondly, the integrator voltage $V_{\text{integrator}}$ is permanently elevated for larger values of V_{syntc} than 1.45 V, there is an upper limit to synaptic time-constants of about $\tau_{syn} \approx 10\,\mu\text{s}$ for neurons with a positive DC-offset. As positive side effect it might help to soften the effect of an negative DC-offset, but this can not be controlled well.

Thirdly, the characteristics of the restive element limits the number of spikes that can be integrated, as for larger integrator voltages $V_{\text{integrator}}$ its strength drops rapidly. The exponential relation between the control parameter V_{syntc} and the resistance R_{syntc} amplifies the uncertainty induced by the floating gates.

Lastly, the integration of synaptic pulses is surprisingly prone to the chosen synaptic time-constant. Only for longer time-constants it is possible to elevate $V_{\text{integrator}}$ over 1.4 V and make use of the whole input range of OTA_1. This leaves a very narrow range before the effect the elevation of integrator voltage $V_{\text{integrator}}$ kicks in. The unexpectedly strong dependency of synaptic weights and V_{syntc} will make a general calibration of the synaptic weights difficult.

In sum these effects will limit the maximal reachable strength of synaptic events



A I_{syn} = 2.4 (——), 4 (– –) and 7.3 µA (·–·–·)
B V_{syntc} = 1.20 (——), 1.30 (– –), 1.40 (·–·–·) and 1.50 V (-----)

Figure 6.7 The plot shows the maximum integrator voltage $V^{max}_{integrator}$ as defined in Figure 6.6. **A** For synaptic pulses with a strength below 10 µA the maximal integrator voltage first rises linearly with the control parameter V_{syntc}. And then rises quickly for $V_{syntc} > 1.5$ V. This roughly reassembles the rise of the resistance R_{vsyntc}. However this cannot be compared with Equation (6.6), because we cannot define a time-constant for the irregular decay of $V_{integrator}$. **B** For the varying pulse strengths the maximal voltage first grows roughly linear as we would expect from Equation (6.6). But for $I_{syn} \gtrsim 7$ µA it begins to fall again. This is caused by the fact, that to strong pulses decay even faster, as shown in Figure 6.4.

and leave only a small range of synaptic time-constants available. This is contrary to the original design goal to provide three orders of magnitude available for the time-constants [Millner 2012]. The membrane time-constant is also affected, as the compensation of DC-offset forces the neurons into an unwanted fast operation mode. This weakens the effect of synaptic events and limits the dynamic range of the neuron membrane as described by Equation (2.7).

6.2 Synaptic Events in the Neurons

We now move forward to the real neurons to evaluate the synaptic time-constants and the actual effect of the DC-offset. We can only measure the membrane potential, as direct measurements on other parts of the neuron are not possible. Therefore, we systematically investigate the effect of synaptic events on the membrane and deduce from these the properties of the synaptic input.

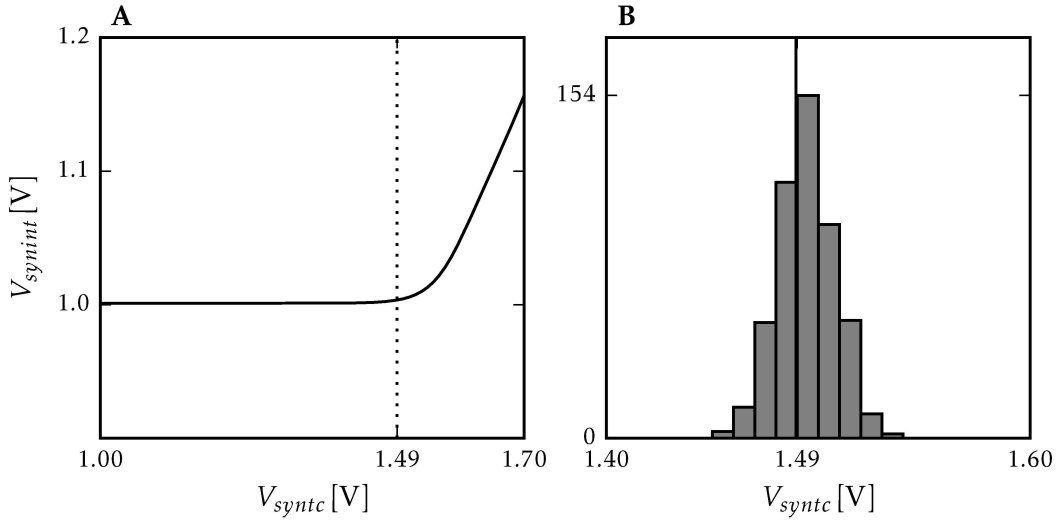


Figure 6.8 A Typical circuit simulation of the synaptic input with a sweep of V_{syntc} without any synaptic input and $V_{\text{syn}} = 1\,\text{V}$. It can be observed that the voltage of the integrator begins to rise at $V_{\text{syntc}} \approx 1500\,\text{mV}$. The vertical line marks a rise of 2 mV of $V_{\text{integrator}}$. **B** Monte Carlo simulation of the sweep ($N = 500$). The histogram shows the V_{syntc} values at which a rise of 2 mV of $V_{\text{integrator}}$ was reached. Both simulations were conducted for $V_{\text{syn}} = 1.0\,\text{V}$. For are an offset of 2 mV the resulting current is about $1/10$ of the one caused by the missing offset correction, shown in Figure 6.1. For larger V_{syntc} the resulting current onto the membrane will rise.

We use PSPs caused by the synaptic events. However, the strength and shape of the PSPs varies largely from not detectable over good to badly disturbed. Therefore, we will first develop a set of criteria to evaluate their quality. Afterwards we characterize the strength of the resulting PSPs and the time-constants we can achieve. Lastly, we discuss the results.

The result shown in this section are from an incremental measurement, as described in Section 4.2.4, with a sweep of the parameter V_{syntc} in the range from 1.15 to 1.6 V in steps of 25 mV. We repeat this measurement for four values within possible spectrum of leakage conductance of $I_{\text{gl}} = 0.1, 0.5$ and $2.5\,\mu\text{A}$ with a scaling of $1:3$ and $I_{\text{gl}} = 2.5\,\mu\text{A}$ with a scaling of $1:1$. We present only the results for the smallest I_{gl} value, because for these the PSP are most prominent, while for the other results we didn't find significant differences. To reduce the noise, we averaged over about 200 PSPs, with the method described in Section 4.3.1. Afterwards we use the fit described in Section 4.3 to determine time-constants and the height of the PSP. Further parameters are listed in Appendix A.4.2.

6.2.1 Evaluation of the Quality

For a robust calibration and measurement, we need to evaluate the quality of the obtained data. The PSPs cannot only be afflicted by faulty circuits, but also by unexpected external events like a bad power supply or interference for example from switching regulators. As we work towards the calibration of millions of neurons, these should be reliably detected to avoid bad calibrations interfering with experiments. We therefore apply three criteria to reject bad measurements.

The first criterion checks if a PSP is presence of an at all. This is an important check, as defect components or faulty configurations can occur at any time. We first used an analysis of the power spectrum for this. But this method is computationally quite costly, so we switch to an faster arbitrary defined metric with equivalent results, but requiring less computational effort. We compare the standard deviation of the averaged recording σ_{rec}, with the σ_{est} is the error estimate as defined in Section 4.3. We define the signal criterion S as

$$S = \sigma_{rec}^2 / \sigma_{est}^2 > 1.5 \tag{6.7}$$

and require, that

$$S > 1.5. \tag{6.8}$$

The standard deviation is a fast calculation, that easily detects the broaden distribution caused by the PSP. However as the membrane distribution is not Gaussian, no further implications must be drawn from it.

The threshold for S is chosen low enough that even PSPs with an amplitude with a height as small as 3 mV are accepted. As we see in Figure 6.9, more than half of the recorded PSPs are rejected by the signal criterion. As we can expect from our simulation results, this happens mostly for low V_{syntc} smaller than 1.4 V. Increasing the limit for S to 3.0 will only reject about 20 more neurons in each step. Here the spike stimulus decays too fast to create a visible PSP. But also for larger V_{syntc} settings about a quarter of the neurons never show a PSP. This is mostly caused by the DC-offset of OTA_1, but also due to unusually weak OTAs.

The second criterion is the χ_{red}^2 of the applied fit. We select PSPs based on the value obtained from the fit. The distribution for this measurement is shown in Figure 4.4 and the results have been discussed in Section 4.3. As shown there, most χ_{red}^2 values are either around 1 or have a value larger than 10. As we want to determine the time-constants from the fit, we choose a limit of

$$\chi_{red}^2 < 1.5. \tag{6.9}$$

From $V_{\text{syntc}} = 1.4\,\text{V}$ on, more neurons are rejected by χ_{red}^2 than by the signal criteria. As Figure 6.9 shows, the number of PSPs rejected by this criteria continuously increases with larger V_{syntc}.

Various reasons reduce the quality of the PSP for larger PSPs : Firstly, as we

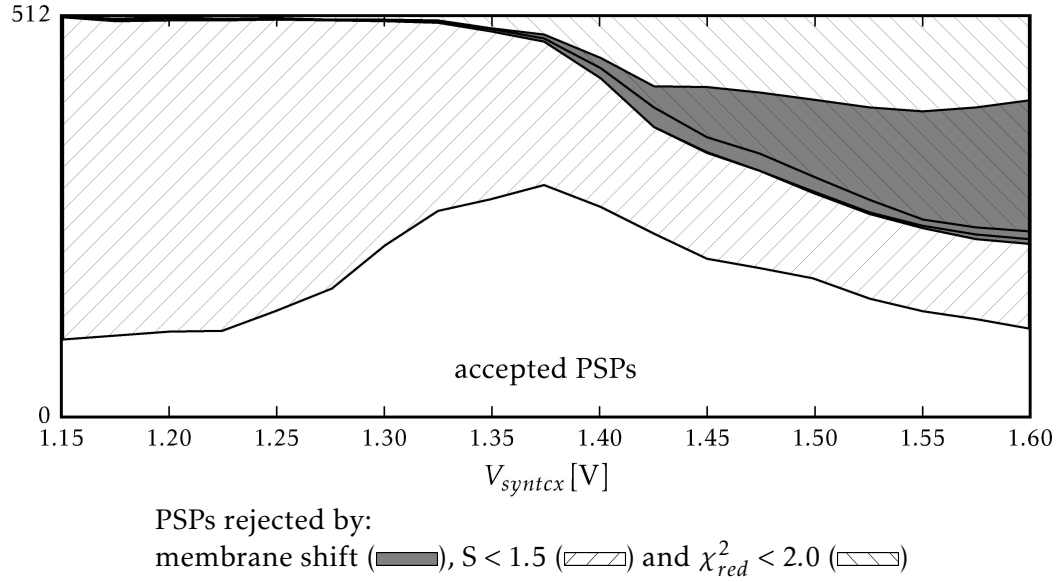


Figure 6.9 The strong variation on HICANN revision 2 forces us to discard PSPs that match one or more of the three criteria defined in Section 6.2.1. Only the measured PSPs in the white area are accepted. In the overlapping areas multiple criteria apply. Note that the signal and the χ^2_{red} criterion never overlap in this data.

saw before, for the decay of the integrator voltage $V_{\text{integrator}}$ no longer follows an exponential. This is a underlying assumption of Equation (2.13). Secondly, OTA_0 saturates, resulting in a flat plateau on the PSP. And lastly, the leakage current from the synaptic input increases, because $V_{\text{integrator}}$ rises at larger V_{syntc}. This violates the assumption, that the PSP originates around the resting potential. All three can also occur at the same time.

We also note that the χ^2_{red} and signal criteria reject PSPs strictly exclusive to each other. The reason is, that the PSP model can perfectly fit a recording without a visible PSP. It will usually find a high out liner in the noise, which may be about 1 mV above the mean, and choose it as maximum for the PSP. Because the recording is almost only noise this will result in fit with χ^2_{red}, but arbitrary time-constants. This shows the importance of the signal criterion.

As the last criterion, we require that the resting potential E_l is not shifted due to large V_{syntc}, as discussed in Section 6.1.3. As threshold we require that the offsets all lay within 15 mV of the lowest value for the excitatory and of the highest for the inhibitory input. As Figure 6.9 shows, this mostly overlap with the χ^2_{red} criteria. However, as this comes along with a even stronger leak current we need to avoid all such cases.

These criteria rule out almost half of the neurons for the best setting of $V_{\text{syntc}} = 1.375\,\text{V}$. Especially for larger V_{syntc} values, which are required for biological plau-

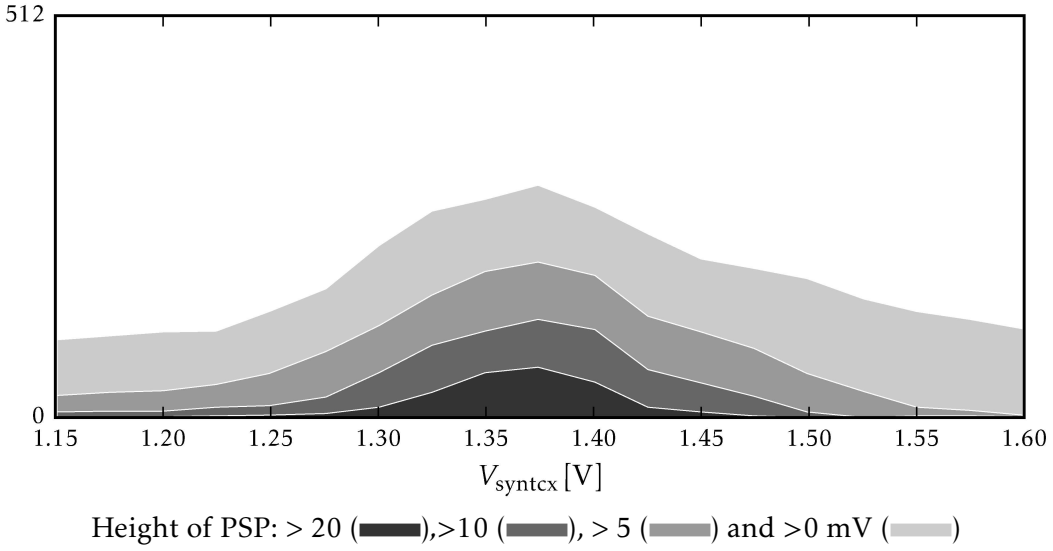


Figure 6.10 Distribution of PSP heights for given V_{syntcx} values. Only accepted PSPs are shown.

sible time-constants as well as for a sufficient maximal voltage in the integrator $V_{\text{integrator}}^{\text{max}}$, the situation is actually worse: Even if we accept badly shaped PSPs with unknown characteristics, only about half of the synaptic inputs remain. Moreover, each neuron has two synaptic inputs, which are both needed to be successful for being fully functional. This decreases the chances that the neuron can conduct as a fully functional neuron even further.

6.2.2 Strength of Postsynaptic Potentials

The measurement also allows a quantitative analysis of the possible impact of an PSP on the membrane. As measurement for the strength of the PSP we are using its height h. Figure 6.10 shows the number of accepted PSPs, that are larger than 0, 5, 10 or 20 mV. The most PSPs are accepted for $V_{\text{syntc}} = 1.375\,\text{V}$, and here we also get the strongest PSPs. For larger V_{syntc} the height and number decreases again. This is caused by the effect, that the PSPs for larger V_{syntc} decay in shape or that E_l is shifted. Therefore, each synaptic input falls out of the distributions after the largest V_{syntc}, that still leads to an accepted PSP.

We therefore determine the maximal possible strength of each input by taking the largest valid PSP produced by it. As shown in Figure 6.11 for the given HICANN 434 neurons can create acceptable PSPs, but half of these are weaker than 10 mV in height. The strongest PSPs are generated for V_{syntc} values around 1.375 V.

We repeated the simulations shown in Section 5.3 for the used set of synaptic parameters and obtained a synaptic pulse of $I_{\text{syn}} = 190\,\text{nA}$, which lead to a rise in the integrator voltage of $\Delta V_{\text{integrator}} = 156\,\text{mV}$. Since such a rise will already push the output current I_{OTA_1} of OTA_1 to half of its maximum value, the PSPs are already

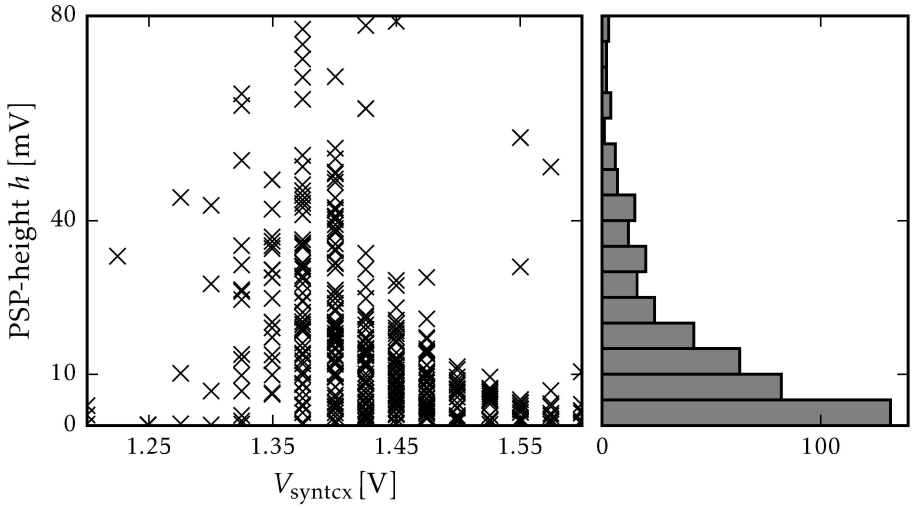


Figure 6.11 The height and V_{syntcx} setting of the strongest possible PSPs for each neuron are shown. 434 neurons can create acceptable PSPs, but the median of the distribution of heights is only 10 mV. Therefore, most of the synaptic inputs are to weak to affect the membrane sufficiently.

at half of the maximal possible strength. For most neurons this means that their maximal reachable deviation from the resting potential remains very small.

6.2.3 Synaptic Time-Constants

The general quality of the PSP is not sufficient to obtain many data points for evaluation. For over 100 neurons on a single HICANN we have less than 4 data points with valid results and only 30 neurons with more than 10 points. For the other neurons most points are settled at $V_{\text{syntc}} < 1.4\,\text{V}$ resulting in very small time-constants. We present an example of the obtained synaptic time-constants in Figure 6.12 together with the corresponding averaged PSPs. We cherry-picke one of the best neurons because for many neurons no PSP at all is visible, while others have only very weak PSPs.

To distinguish the time-constants we use the small setting of $I_{\text{gl}} = 100\,\mu\text{A}$, which leads to long membrane time-constant. We therefore assume that the smaller time-constant, obtained by the fit, is τ_{syn}. However, the minimum value of τ_m is also influenced by minimum synaptic conductance caused by the DC-offset in the synaptic input. This and the large possible range of values, makes is generally difficult to distinguish the time-constants. Further the fit often decays for larger V_{syntc} and both time-constants become equal. The values obtained by the fit often indicate an exponential relation between V_{syntc} and τ_{syn} as intended, but with the given number of data points per neuron and the variation in the data, a comparison to the model did not seem feasible. The average values over all neurons also indicates

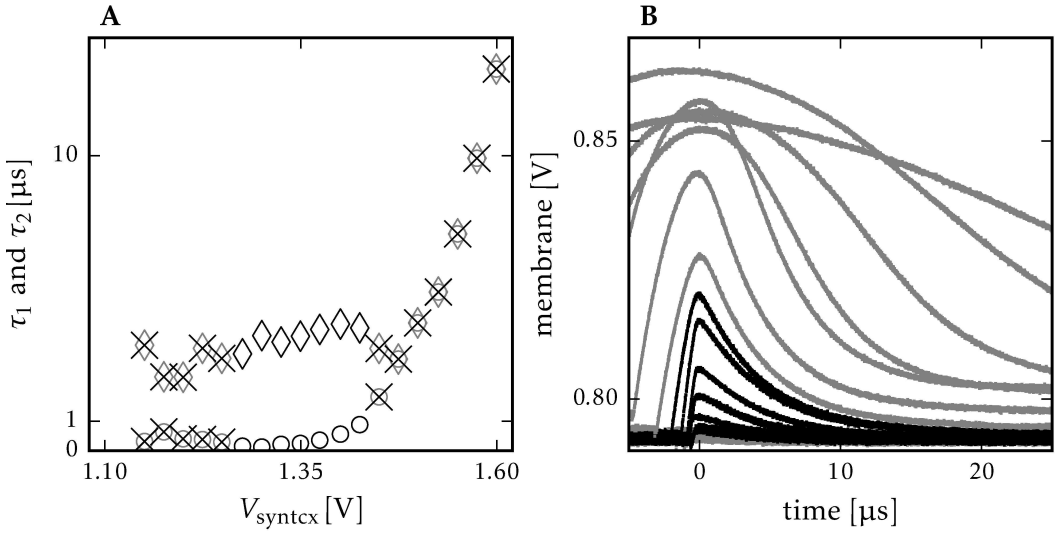


Figure 6.12 A Results of the time-constant fit for an exemplary good neuron: Both time-constants τ_1 (○ ○) and τ_2 (◊ ◊) are shown. The crossed values were discarded, because of their bad quality. **B** The averaged traces to all measured values of V_{syntcx}, discarded traces are gray.
Because the membrane is set to a very slow value, we can identify the smaller time-constant τ_1 as τ_{syn}. This plot also shows, that also the membrane time-constants can vary. This is most likely cause by the variation of the fit. Note, that the selected neuron is not representative, most show worse results.

an exponential relation for V_{syntc} from 1.25 to 1.5 V, but with large errors.

Finally, we determine the time-constants that were reachable by the accepted PSPs. As shown in Figure 6.13, we can obtain for $V_{\text{syntc}} = 1.4\,\text{V}$ time-constants of $\tau_{\text{syn}} \approx 1\,\mu\text{s}$ and for $V_{\text{syntc}} = 1.6\,\text{V}$ of $\tau_{\text{syn}} \approx 10\,\mu\text{s}$. However, there are only 268 or rather 113 neurons, that have good PSPs for these settings.

Lastly, we want to remark that a long time-constant does not correlate with a strong PSP. The correlation plot is also shown in Figure 6.13. In general a raising time-constant should always cause a larger PSP, but the distortions form the DC-offsets and the limited linear range of the OTA are too disturbing in this case.

6.2.4 Summary

To sum up, we can say that the DC-offset of OTA_1 prevents half the synaptic inputs from generating PSPs. As each neuron has two synaptic inputs which have to generate PSPs successfully, this lowers the chance for getting working neurons severely. But even for the generating synaptic inputs the effective membrane time-constant is driven up by the minimal synaptic conductance. We see this in the low number of neurons that are able to produce larger PSPs.

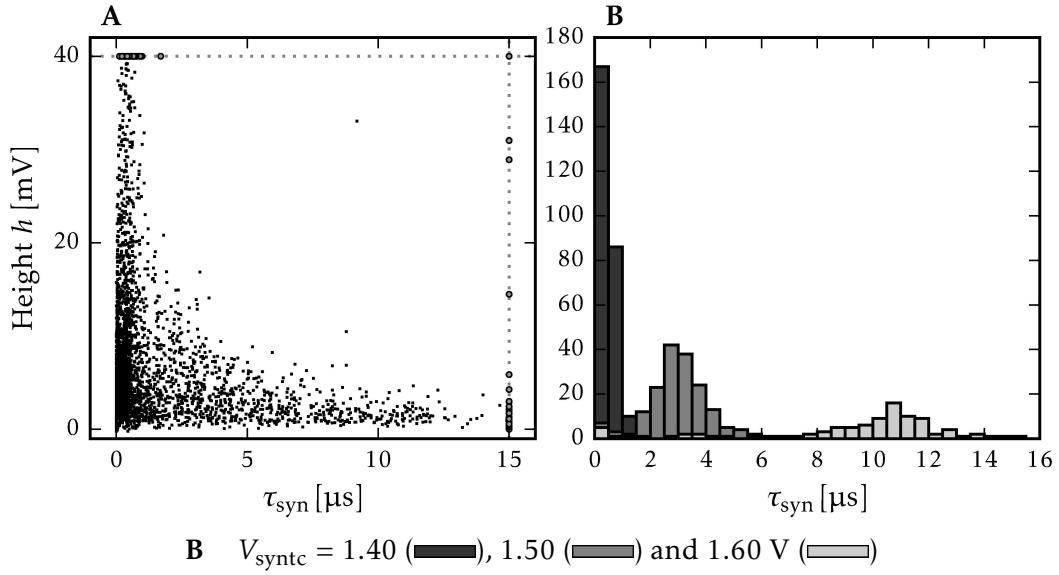


Figure 6.13 A Accepted time-constants τ_1 of all neurons. We can identify τ_1 as τ_{syn}, because leakage conductance is set very low, resulting in long membrane time-constant τ_m. The outliers with $h > 40\,\text{mV}$ or $\tau_{\text{syn}} > 15\,\mu\text{s}$ are plotted at the boarder. **B** Available time-constants for different settings of V_{syntc}. The total number of available neurons is 268, 177 and 113.

In addition, the restive element of the synaptic input has quite unfavorable characteristics: The exponential dependency is concentrated in a far to small range of its control parameter V_{syntc}. Its instability against voltage differences worsens the PSPs further.

The sum of all these effects cause each synaptic input to have a small range of V_{syntc}, which spans only over 0.1 to 0.2 V, where it might produce acceptable PSPs. For about 70 % of the synaptic inputs the generated PSPs are weaker than 10 mV or even not existing for the given parameters. The remaining PSP are not suitable to develop a good calibration for the time-constant τ_{syn}. Lastly most of the synaptic inputs are too weak, compared to the leakage conductance, causing generally quite weak PSPs.

6.3 A Functional Calibration

The given state of the synaptic input requires a pragmatic calibration approach. We have to counter-balance the opposing effects acting on the neuron membrane. The DC-mismatch calls for a strong membrane conductance g_{l} to compensate the leakage currents from the synaptic inputs. Otherwise, the calibration of the resting potential deviates vastly from its target E_{l} value.

On the other hand a large g_{l} reduces the maximum V_{min} or maximal membrane

potential V_{max} of the neuron. We can modify Equation (2.7) to give us expressions for both values:

$$V_{\text{min}} = \frac{g_i^{max} E_{\text{syni}} + g_{\text{l}} E_l}{g_i^{max} + g_{\text{l}}} \text{ for } g_x = 0 \text{ and} \tag{6.10}$$

$$V_{\text{max}} = \frac{g_x^{max} E_{\text{synx}} + g_{\text{l}} E_l}{g_x^{max} + g_{\text{l}}} \text{ for } g_i = 0, \tag{6.11}$$

where g_i^{max} or g_x^{max} are maximal possible conductances of the given synaptic input. For larger g_{l} both values will be shifted towards E_l, reducing the maximum dynamic range the membrane can cover. This makes it more difficult to place the spike threshold reliably between resting and reversal potential, in a way that the membrane is capable of triggering a spike. Especially under the circumstances, that most inputs only produce very weak PSPs. This motivates us to select g_{l} as small as possible.

The following calibration method tries to balance these effects without introducing an overly complex and time-consuming amount of measurements. We developed it in close cooperation with Mitja Kleider, Dominik Schmidt and Sebastian Schmitt and it was shown in similar form in the Masterthesis of [Schmidt 2014]. To handle the complex interactions between the synaptic inputs and the neuron, we choose a fixed set of target values for the calibrations. For the resting potential $V_{\text{target}} = 0.8\,\text{V}$, and the reversal potential symmetrically around it at $E_{\text{syni}} = 0.65\,\text{V}$ and $E_{\text{synx}} = 0.95\,\text{V}$. First we do the regular calibration for the readout shift and all potentials [Schmidt 2014], except for the resting potential E_{l}. Afterwards we search a fixed membrane leakage I_{gl}, then optimize V_{syntc} on both inputs to get acceptable PSPs and lastly search an E_{l} value that gives the desired resting potential. However, it is easily possible to adapt the parameters to your own specific needs.

6.3.1 Selecting Membrane Conductance

We start selecting of the membrane time-constant. Our goal is it to find the smallest possible I_{gl} value, that is strong enough to counteract the leaky synaptic inputs. For this, we conduct for a series of measurements with increasing I_{gl} values. For each I_{gl} value we determine the membrane potential for the settings of E_{l}.

Additionally we obtain the minimal reachable membrane voltage V_{mem}^- and the maximal reachable V_{mem}^+ for these parameters. We then choose the lowest value of I_{gl}, where conditions

$$V_{\text{mem}}^- < V_{\text{target}} - 50\,\text{mV} \text{ and } V_{\text{mem}}^+ < V_{\text{target}} + 50\,\text{mV} \tag{6.12}$$

are fulfilled. This choice allows us to have a minimal I_{gl} and sufficient room to work around trial to trial variations.

We found, that I_{gl} values of 0.1, 0.2, 0.3, 0.6, 1.2 and 2.3 µA give a good balance between the number of steps and the reached precision. Note, that the relation

between I_{gl} and g_l is not linear [Schmidt 2014]. For E_l we use the same values as for the reversal potential 0.65 and 0.95 V. Choosing a smaller range will generally lead to larger I_{gl} values.

6.3.2 Selecting Strongest Possible PSP

The next calibration step we want to select PSP has the strongest possible effect. This compensates the missing direct time-constant calibration and ensures that the input has an maximal effect on the neuron. From our measurements shown in Section 6.2, we know that the total height h of the PSP correlates well with the standard deviation of the recorded membrane. On the other hand, as shown in Figure 6.13, the time-constant τ_{syn} does not correlate with the height h. Therefore we can stimulate the neuron with regular spikes and then use the standard deviation as fast and simple measure to choose an optimal V_{syntc}, that leads to the strongest possible PSP for a neuron. The only further condition is, that the input rate has to be low enough that PSP will not overlap. Also we can omit the computationally intensive step of averaging the PSPs.

Additionally, we found, that we do not need to check if the resting potential E_l is shifted for larger V_{syntc} values, as described in Section 6.2, . Our evaluation shows only very little differences when such a check was included.

Figure 6.14 shows the finally chosen V_{syntc} values. We use ten V_{syntc} steps from 1.35 to 1.70 V and a spike stimulus with 25 kHz, $g_{div} = 2$ and $w = 15$. For each neuron we record the membrane for about 600 µs, which is sufficient to capture 15 spikes.

6.3.3 Selecting Resting Potential

The previous calibration of the reset potential gets invalidated by the previous two steps, because both influence the conductance of the membrane and the synaptic inputs in steady state. With the chosen values for I_{gl}, V_{syntci} and V_{syntcx}, their conductance now only differes due to trial-to-trial variability. Subsequently we can now shift E_l to a value, where the membrane conductance compensates the minimal conductance of the synaptic inputs to reach the desired resting potential V_{target}.

We recalibrate E_l by repeating the regular E_l calibration step with the fixed values I_{gl}, V_{syntci} and V_{syntcx}. Figure 6.15 shows the distribution of the resting membrane potentials before and after the recalibration step. Before recalibrating, the membrane is drawn to the reversal potentials. This is a good example, how the DC-offset of OTA_1, shown in Section 6.1, affects the calibrations: After recalibrating, the membrane is distributed around $V_{mem} = 799 \pm 11$ mV using the standard deviation as error. We reach the desired target value of $V_{target} = 800$ mV, but the error is larger than we can expect from the previous calibrations of the potentials. This is expected because results of the previous steps also depend on the chosen value of E_l. This could be improved by either a repeated calibrations cycle or by trying to optimize all parameters in one step. However, both variants would increase the required time

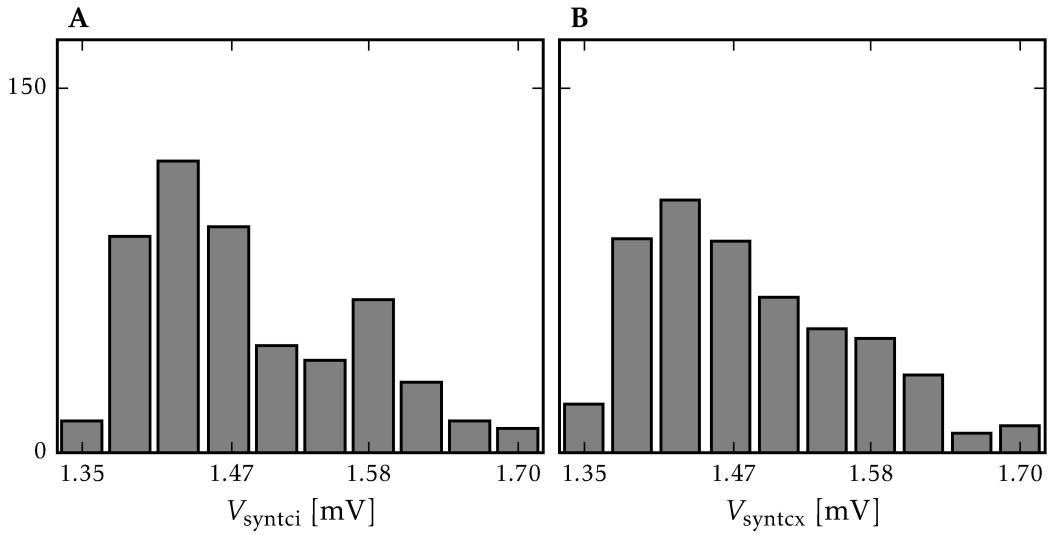


Figure 6.14 Result of the V_{syntc} calibration. For each neuron the V_{syntc} value is chosen, that causes the strongest standard deviation of the trace, while not yet causes yet a rise in the membrane voltage. The plot shows the number of neurons for **A** inhibitory and **B** excitatory synaptic inputs.

drastically and also we cannot predict the best possible precision reachable. So our method gives, so far, the best trade-off between precision and required time.

6.3.4 Evaluating the Calibrations with Spike Input

Lastly, we want to evaluate how good the synaptic input can react to spike input. To estimate this effect we applied the best calibration parameters to the neurons, and then recorded the membrane in the resting state V_{rest}, under inhibitory stimulus V_{min} and under excitatory stimulus V_{max}. The difference $V_{\text{max}} - V_{\text{min}}$ describes the maximal dynamic range the membrane of a neuron can cover. The stimuli are changed without reprogramming the floating gates. We use four background generators with a rate of 5 MHz, a synaptic weight of $w = 15$ and a divisor of $g_{\text{div}} = 2$.

The results are shown in Figure 6.16. The distribution of V_{min} and V_{max} are mirrored symmetrically around the target potential of $V_{\text{target}} = 0.8\,\text{mV}$. Further the Pearson pairwise correlation coefficient between V_{min} and V_{max} is -0.1. These observations confirm our basic assumption that the inhibitory and excitatory inputs have an equal behavior and influence the neurons independently from each other.

For an ideal synaptic input we would expect that both distributions are centered in between the resting and reversal potential. The exact location is given by Equation (6.11) and depends on the maximal reachable conductances. However, the measurement shows that the excitatory input cannot shift the membrane of half

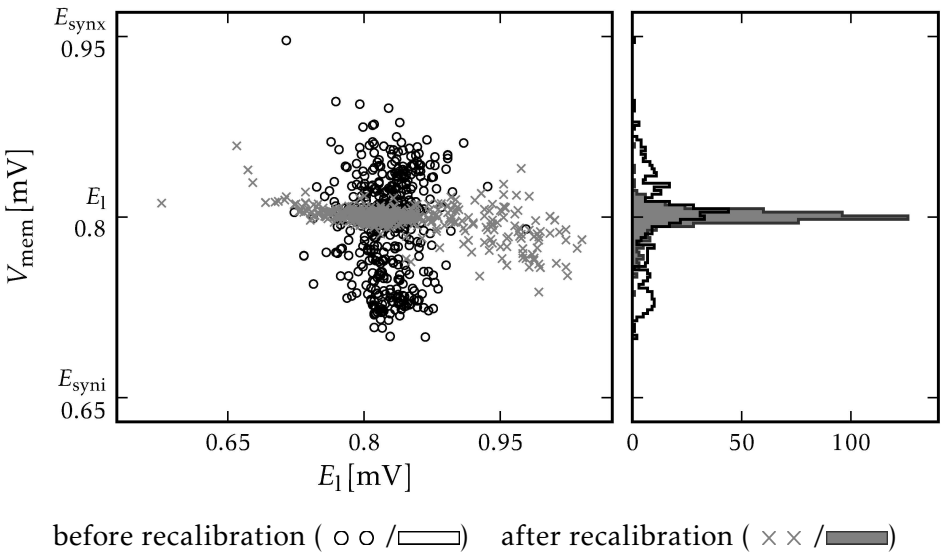


Figure 6.15 Measurement of the resting voltage before an after recalibration. The recalibration can counteract the spread induced by the leakage from the synaptic inputs. The recalibrated distribution is centered around 799 ± 11 mV (standard deviation as error). This is about the double, as what we can obtain for the other potentials. After recalibration more extreme E_l values are required. This is required to compensate for the minimal conductances of the synaptic input caused by the DC-offset of OTA_1.

of the neurons over 0.82 V and for 75 % not over 0.84 V. For the inhibitory input the situation is similar, it shifts the membrane of half of the neurons below 0.77 V and for 75 % not below 0.75 V. We anticipated this in Section 6.2.2, but here we actually max the synaptic input out. These values are quite sensitive to trial-to-trial variations. Only about a quarter of the neurons can cover a dynamic range larger than 0.1 mV. The mean of its distribution is at about 0.05 V.

We have to set these results in the context of the trial-to-trial variability for each parameter. It is required to place inhibitory, resting, threshold and excitatory potential in this order raising. Therefore, we tested the trial-to-trial variability for resting potential by reconfiguring the HICANN 20 times with the given parameters from the calibration and read out the resting potential. The results are shown in Figure 6.17.

As we expect from the direct measurements of the floating gate cells, shown in Section 5.1, the variability fluctuates strongly between the neurons. The average variation per trial has a standard deviation of about 5 mV for each neurons. However, the worst neuron has a variation of 18 mV. This makes is very difficult for most of the neurons to find a threshold that will not cause permanent spiking, because on one hand it might fall below the resting potential and on the other hand it might be so high that the neurons can not reach its.

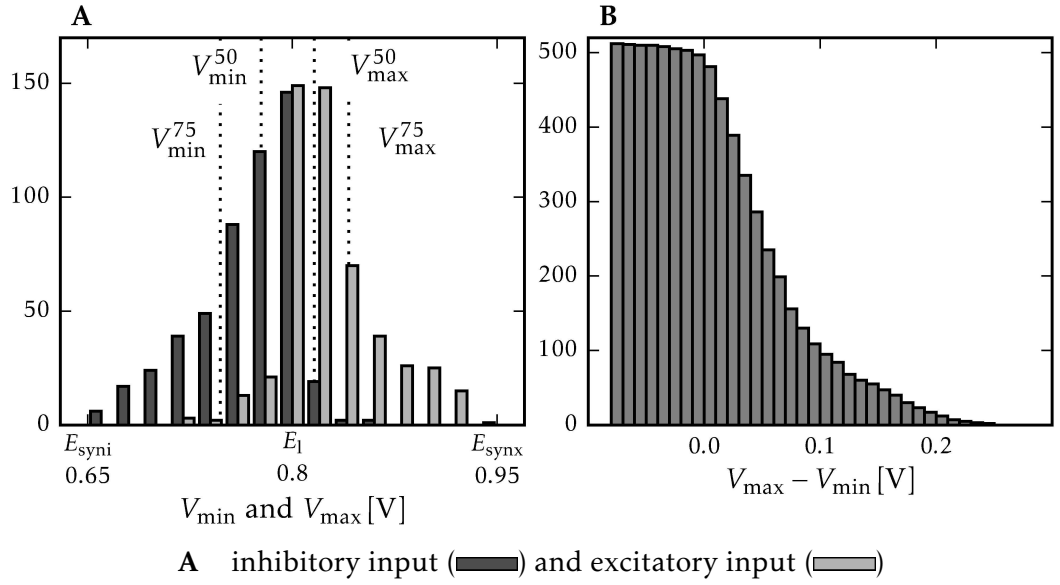


Figure 6.16 Measurements of the maximal response to spikes. The neuron is stimulated using four background generators with a rate of 5 MHz, weight $w = 15$ and $g_{\text{div}} = 2$. **A** Distribution of the lowest reachable membrane voltage V_{min} for spike stimulus from the inhibitory input and of the highest reachable membrane voltage V_{max} for spike stimulus from the excitatory input over all neurons. The symmetry of both distribution confirms that both synaptic inputs have identical properties. We confirmed that there is no correlation between V_{min} and V_{max}. The resting potential is distributed as shown in Figure 6.15. The histogram uses 0.02 V bins. The 50 % and 50 % quantiles (from E_{l}) are: $V_{\text{max}}^{50} = 0.816$, $V_{\text{max}}^{75} = 0.841$, $V_{\text{min}}^{50} = 0.777$ and $V_{\text{min}}^{75} = 0.747$ V.
B Cumulative distribution for the maximal dynamic range $V_{\text{max}} - V_{\text{min}}$ over all neurons. The small number of neurons, that have a large range, combined with the higher variation in setting E_{l} makes it problematic to find neurons, that have a good response to spike input.

We further conducted some measurements to explore the possibility to increase the distance between reversal potentials. Firstly, the effects shown before will scale, due to the conductance based synaptic input. At some point we will then leave the linear range of the OTAs, which makes it more difficult to compensate for the minimal conductances of both synaptic inputs. Also the membrane dynamics becomes impaired at this point. For measurements with $E_{\text{syni}} = 0.5$ V and $E_{\text{synx}} = 1.1$ V, we found a membrane distribution after the E_{l} recalibration of $V_{\text{mem}} = 802 \pm 27$ mV. This is more than twice what we found before and even more extreme E_l setting are required to compensate for the leakage of the synaptic inputs. The distribution of the dynamic input range will get a factor of about two broader. So lastly, the larger uncertainties negate the advantages of a higher dynamic range and

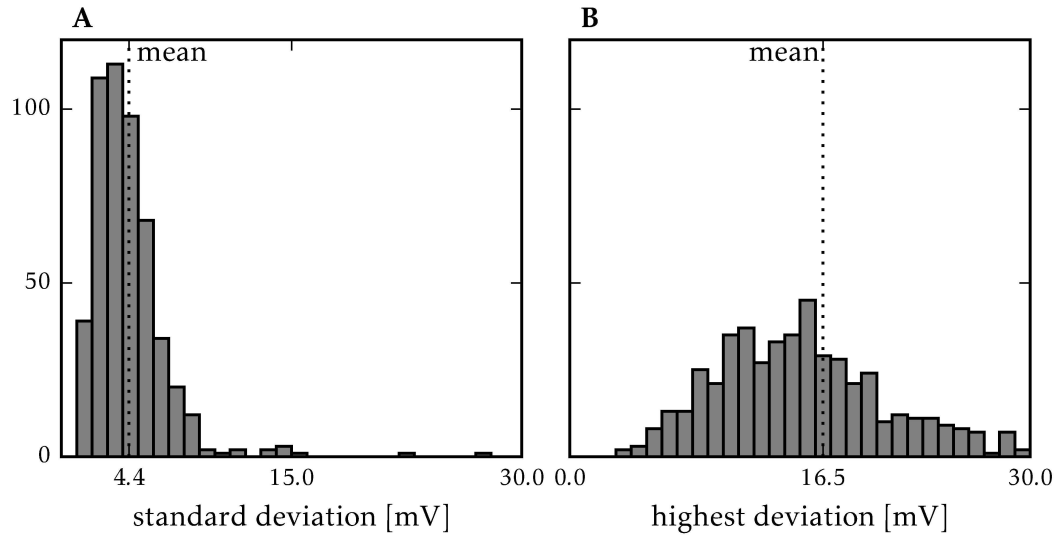


Figure 6.17 Measurement of the try-to-try variations for the resting voltage with fixed calibration for I_{gl}, E_{syni}, E_{synx} and V_{syntci} and V_{syntcx} as described in Section 6.3 for $E_l = 0.8\,V$. The histograms shows the distribution of the **A** standard deviation and **B** difference between highest and lowest values for each neuron. 20 measurements were done. 29 values larger than 30 mV were omitted.

one has to find a good trade-off depended on the actual use-case.

Even with the given quality of the neurons simpler network experiments are possible. Figure 6.18 shows a simple feed forward neuron chain done by Schmidt [2014]. However we observe than most of the neurons actually do not react to spikes. This is compensated by using increased populations and larger redundancy.

6.3.5 Calibration Speed

The whole calibration shown here currently takes about 25 min. It consists of the calibration of the readout shifts, V_{reset}, V_t, E_{syni}, E_{synx}, I_{gl}, E_l, V_{syntci}, V_{syntcx} and a second time E_l. The most time consuming steps are the calibration of V_{syntci} and V_{syntcx} each taking about 5 min. These are just approximate numbers for giving the reader a basic impression of the required time scale. We will follow up in this matter in the results for revision 4 in Section 7.3.5.

6.3.6 Summary

The restriction of the synaptic input in HICANN revision 2 is now fully immanent. The strong interaction of the control parameters and the DC-offset of the synaptic make a calibration of the parameters independently from each other, as proposed by Schwartz [2013], not feasible. Further obstacles are both the overly strong synaptic

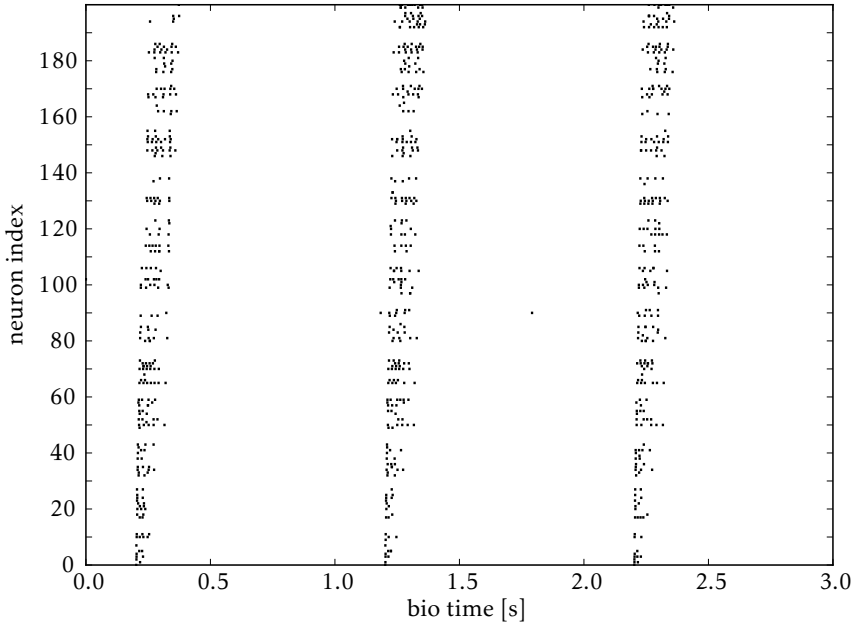


Figure 6.18 Rasterplot of a feed-forward neural network emulation with 200 neurons across two calibrated HICANN chips. Spikes are sent to population 0 (located on the lower end) in one second intervals and propagate upwards through the chain. Figure and caption taken from [Schmidt 2014]

events, which easily saturate the synaptic input, as well as that the observable synaptic conductances are too weak compared to the leakage conductance. This limits the dynamic range the neurons can use.

Even with the best effort calibration we presented, most neurons will only have limited capability to be used in neuronal network experiments. In combination with the given precision of the floating gates the dynamic range of their membrane is not high enough to be reliable usable. Examples of the effort required to get a functional subset of the neurons can be found in the works of Alevi [2015], Nonnenmacher [2015] and Schmidt [2014]. Based on the results presented in this chapter we recommended a new revision of HICANN.

7 Synaptic Input on HICANN Revision 4

In this chapter, we present the redesigned synaptic input of HICANN revision 4 and 4.1. While the original concept by Millner [2012] still remains in use, some of its components needed to be improved. These were designed by Schemmel [2015]. In Section 7.1 we present these changes in greater detail and show the characteristics of the new circuit elements. Afterwards we repeat the simulations of the integration of the synaptic events and find that the changes lead to the desired behavior, for example a large number of spikes can now be integrated and the decay of the time-constant is exponential. In Section 7.2 we show the calibration for the compensation of the DC-offset of OTA_1. We find, that the calibration leads to good results, and present an outlook on how the number of needed calibration steps can be reduced. Finally, we present in Section 7.3 the calibration of the synaptic time-constants. We show that the method we propose can also be used to calibrate the membrane time-constants. A final evaluation of both synaptic and membrane time-constants shows that we can reduce the spread of the parameter distribution. We conclude that the HICANN calibration is ready for experiments using the LIF neuron model.

7.1 Synaptic Input

While we already briefly discussed the changes of the synaptic input in Section 2.2.1, we here describe the circuits in greater detail, show their relevant characteristics and highlight the improvements towards HICANN revision 2. However, these improvements come at the cost of larger neuron size and hence four synapses per neurons needed to be removed to free the required area on the chip. Afterwards we repeat the simulations evaluating the integration of spike events for the settings shown in Section 6.1.2. We then reevaluate the technical limits for the control parameter V_{syntc} in Section 7.1.3. We close the section with a summary.

7.1.1 Changes in the Synaptic Input

We accompanied the design process – in close cooperation with Mitja Kleider and Paul Müller – by testing the integration of the new circuits into the neuron. For this task we created the integration of circuit simulations into the calibration software, as described in Section 4.4.4, with both typical and with Monte Carlo simulations. This led to various iterations of designs. The success of the early tests during the design phase is reflected in the calibration results shown in Sections 7.2 and 7.3.

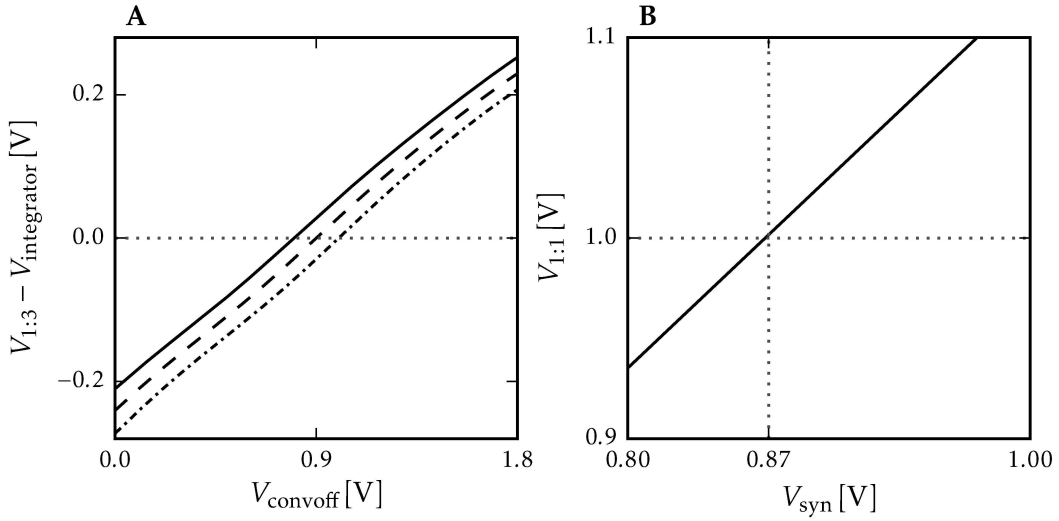


Figure 7.1 The simulation shows the characteristics of the bias generator used to compensate the mismatch of OTA_1 in the synaptic input.
A Effect of the control parameter V_{convoff} on the reference voltage for OTA_1 for $V_{\text{syn}} = 0.8$ (——), 0.9 (– –) and 1.0 V (·–·–·). For $V_{\text{syn}} = 0.9\,\text{V}$ the mean transformation factor from V_{convoff} to the difference is 0.28. **B** The reference voltage V_{syn} gets also shifted by the voltage divider. The plots show the relation between the control parameter V_{syn} and the actual reference voltage of the integrator $V_{1:1}$. To reach the desired value of 1.0 V we need to set $V_{\text{syn}} = 0.87\,\text{V}$.

Offset Compensation

First, a compensation for the DC-offset of the OTA described in Section 6.1 was added. For this, two so far unused voltage parameters were connected to the neurons, one for each synaptic input. The new parameter is named V_{convoff}.

The direct approach would have been to directly connect them as reference for OTA_1. But then only about 0.2 V of the parameter range would habe been used to compensate the offset because this is the total spread of the DC-offset distribution as shown in Figure 6.1. In consequence the offset compensation would be vulnerable to try-to-try variations of the new parameter.

This can be circumvented by rescaling V_{convoff} and then use it as reference for the integrator. This is done by the new bias voltage generator element, that scales V_{convoff} relative to the reference voltage of the integrator V_{syn}. The bias voltage generator has two inputs V_{syn} and V_{convoff} and generates two output voltages: First, $V_{1:3}$, which is connected to OTA_1, and second, $V_{1:1}$, which is used as reference for the integrator. This second conversion is intended to compensate possible process corner variations, that could shift the reference and the compensated voltage too far apart.

The conversion characteristics for both are shown in Figure 7.1. For $V_{1:3}$ we obtain

$$V_{1:3} = V_{\text{syn}} \cdot \alpha \left(V_{\text{convoff}} - V_{\text{syn}} \right) \text{ with } \alpha \approx 0.28, \tag{7.1}$$

where we determined α from the simulation. Whereas, the reference voltage is only shifted by the bias generator and we obtain

$$V_{1:1} = V_{\text{syn}} + \beta \quad \text{with} \quad \beta \approx 0.13\,\text{V}, \tag{7.2}$$

where we determined β from the simulation. Subsequently, we reduce the default setting for V_{syn} to 0.9 V, so that we still have about 1 V as reference for the integrator.

Integrator

The second change affects the integrator of the synaptic input itself. The original resistor in the integrator was designed to support multiple magnitudes of time-constants, but only for a very small input range of 0.2 V of the control parameter V_{syntc} [Millner 2012]. The redesign was used to improve the resistor to make it less prone to try-to-try variations of V_{syntc}. The characteristics of the changed resistor are shown in Figure 7.2.

The resistance R_{syntc} is now less sensitive to the voltage $V_{\text{integrator}}$ in the integrator. We show in Section 7.1.2 that the decay of $V_{\text{integrator}}$ is now indeed almost exponential. Note that the polarity of the parameter is now reversed, therefore the resistance falls for larger V_{syntc}. At first, the relation between V_{syntc} and R_{syntc} is exponential for $V_{\text{syntc}} < 0.4\,\text{V}$, and for larger V_{syntc} the slope becomes smaller and it almost converges to a constant value. Between 0.4 to 1.8 V of V_{syntc}, it covers a resistance R_{syntc} from about 6 to 0.4 MΩ.

In view of the required calibration of the synaptic time-constant, we search a model for the dependency of R_{syntc} on the control parameter V_{syntc}. The transition from exponential rise to the constant value can be described by using a function known as softplus [Dugas et al. 2001]. Because the slope of the relation does not becomes zero for larger V_{syntc}, we need to add a linear correction term. We use as model for the logarithm of the resistance

$$\log R_{\text{syntc}} = \frac{a}{c} \cdot \log\left(1 + \exp\left(c \cdot \left(V_{\text{syntc}} - b\right)\right)\right) + d \cdot V_{\text{syntc}} + R_0, \tag{7.3}$$

where we determine a, b, c, d and R_0 by using a fit against the simulation of R_{syntc}, shown in Figure 7.2. The obtained parameters a listed in Table 7.1

Both additions – the bias generator as well as changing the resistor – increased the required area of the neuron. This allowed to increase the integrator capacitor as well, because it is placed in a layer above those circuits. The capacity is now 471 pF. This reduces the impact of each spike event on the integrator and hence more spikes can now be accumulated.

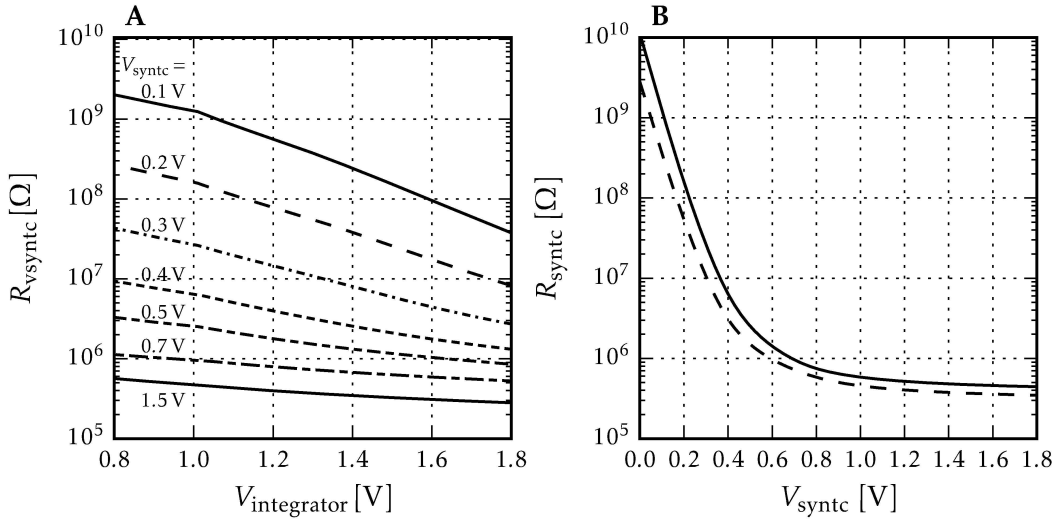


Figure 7.2 A Simulation oft the resistive element in the synaptic input for seven different values of V_{syntc}. The integrator voltage $V_{\text{integrator}}$ is varied, while the other terminal is held at $V_{1:1} = 1.0\,\text{V}$. The stability against voltage changes in the integrator, was improved compared to revision 2. This allows much larger voltages in the integrator and makes the integration of pulses less dependent on V_{syntc}. **B** Resistance depends on the control parameter V_{syntc} for two $V_{\text{integrator}}$ levels. After a slow onset, R_{vsyntc} rises exponentially with lower controlling voltage V_{syntc}.

Scaling of the Synaptic Strength

The third change is strictly spoken not a change in the neuron, but in the analog parameter storage. Only for revision 4.1, the scaling of the reference current V_{gmax} for the synaptic pulses was reduced by a factor 10 to $g_{\text{scale}} = 0.4$. We provided the simulations shown in Section 5.3 to motivate this change. It compensates for the offset in the synapses and allows us to use now a larger range of V_{gmax}.

a	b	c	d	R_0
-25.6	0.364	-6.45	-0.313	13.5

Table 7.1 Parameters obtained by fitting Equation (7.3) to the resistance for an integrator voltage of $V_{\text{integrator}} = 1.01\,\text{V}$.

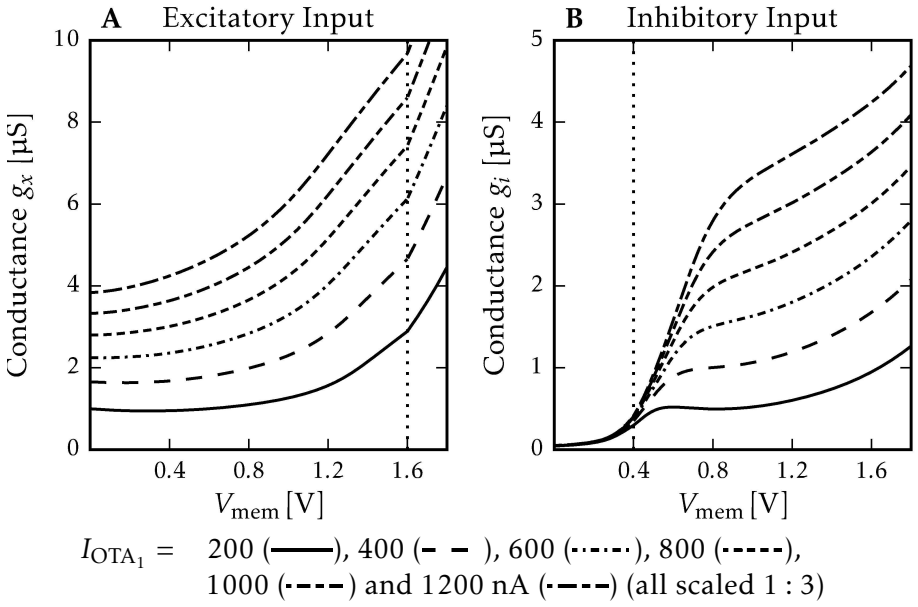


Figure 7.3 Typical DC-Simulation of the synaptic input resistor R_0 used HICANN revision 4.1. The reversal potential is fixed at **A** $E_{\text{syni}} = 0.4\,\text{V}$ and **B** $E_{\text{synx}} = 1.6\,\text{V}$.
Compared to OTA_1 used in the previous HICANN revision, the resulting conductance is no longer symmetric. It gets smaller for smaller membrane voltages V_{mem}. Also the excitatory input is now potentially stronger than the inhibitory input depending on the chosen reversal potentials.

Resistor for the Reversal Potential

Lastly, OTA_0 was replaced by a current controlled resistor. As shown in Figures 2.4 and 7.3, the conductance of this resistor is more stable over a large voltage range than the previously used OTA. This allows to place the reversal potential in a much larger distance to the resting potential than before and so so increases the dynamic range the membrane can cover. This has two beneficial effects: the resting, threshold and reversal potentials of the neuron can now be placed further apart reducing the impact of try-to-try variations; further a more biological realistic distance between those potentials can be used, especially for the excitatory reversal potential. The conductance is now asymmetric for positive and negative voltages differences. Therefore, both synaptic inputs behave now differently.

In contrast to the OTA used in revision 2, the new design pulls the current flowing onto the membrane directly from the voltage floating gate cell. The cell can only provide a sufficient power by setting the bias of the source follower in the cells to its smallest value. Otherwise, the voltage difference over the resistor can drop when the conductance of the synaptic input reaches strong values.

We need to take special care that this will not occur during programming the

floating gate cell, because this would affect the feedback loop of the controller and lead to unpredictable values of the reversal potentials E_{syn}. We can easily avoid this by disabling the synaptic input during the programming of E_{syn}. Therefore, the programming routine of the floating gate was updated: it first sets the bias current I_{conv} of OTA_1 to zero, then programs the reversal potential and afterwards restores the I_{conv} value. Further we want to note, that the int_op bias, which effects multiple other parameters [SP9 Spec 2015], is stored as shared parameter in the same row of the parameter storage as I_{convi} and therefore needs to be programmed at the same time. This has also been taken into account to avoid a negative impact on other parameters.

7.1.2 Integration of Synaptic Events

Most of the changes affect the integration of synaptic events in the synaptic input. Therefore, we repeat the simulations developed for revision 2 and shown in Section 6.1.2, for the updated integrator circuit. We use the simplified neuron simulation with synapses, as described in Section 4.4.3 and simulate the effect of single synaptic pulses. To cover a wide range of possible pulse strengths I_{syn}, we used 15 logarithmically distributed values V_{gmax} from 0.01 to 2.5 µA, using a weight $w = 15$ and a divider of $g_{\text{div}} = 2$. This leads to a synaptic pulse strength of about 2 to 11 µA. For each of the V_{gmax} values, we simulated 27 V_{syntc} settings, from 0.15 to 0.5 V in steps of 0.025 V and from 0.5 to 1.7 V in steps of 0.1 V, taking the exponential relation to the resistance R_{syntc} into account. The length of the synaptic pulse is $t_{\text{syn}} = 10$ ns.

Single Spikes

Figure 7.4 shows the effect of single spikes and their decay in the integrator for exemplary settings. Because of the finite gain of the OP, the integration of the spike takes 50 to 70 ns, a bit longer for higher pulses. For comparison, this is about the time that a spike emitted by the neurons needs to pass through the merger tree and to reach the on-chip bus. Further, the integrator reacts with a short undershoot to the onset of the spike and with an equally short overshoot to the end of the spike. These can be observed as small peaks on the shown traces . They are caused by the frequency response characteristics of the OP, which show a finite reaction time on the voltage changes on its input. The peaks are much shorter than the synaptic time-constants and will not affect the neuron.

We can now compare the behavior of the integrator to its ideal description, as we did in Section 6.1.2. The ideal behavior of the circuit did not change compared to revision 2. However, the small peak forbids that we simply take the maximum to determine the rise in the integrator voltage $\Delta V_{\text{integrator}}$. But the improved resistor allows us to determine $\Delta V_{\text{integrator}}$ and also the time-constant τ_{syn} by fitting an

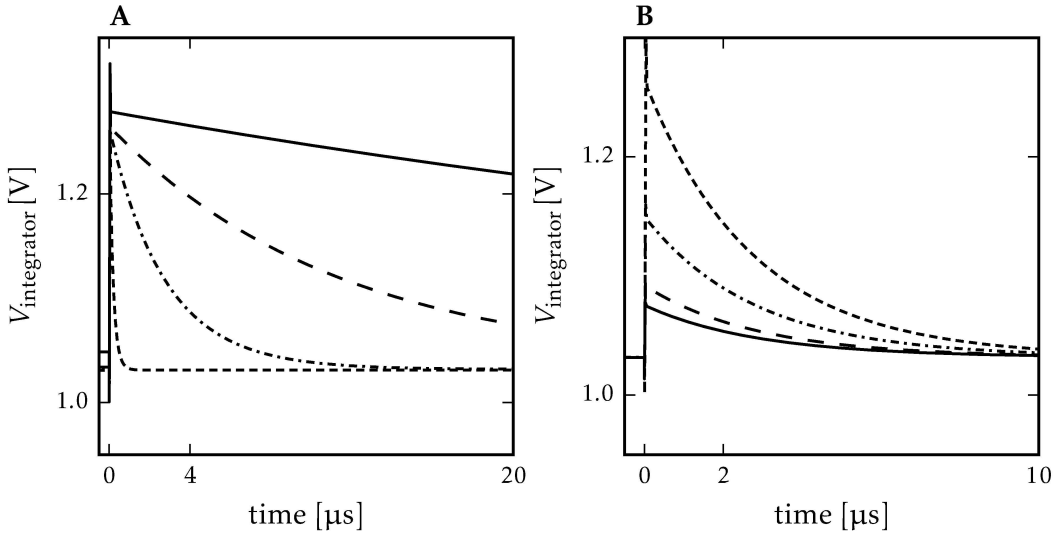


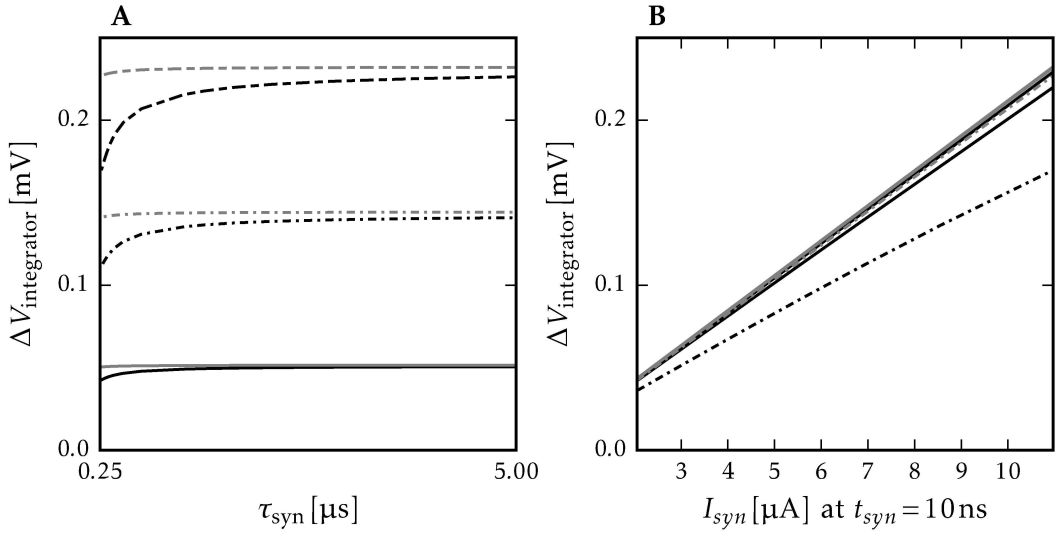
Figure 7.4 Exemplary simulations of the response in the integrator for synaptic current pulses of $t_{syn} = 10\,\text{ns}$. The small peaks at the onset and on top of the spike are caused by the frequency response of the OP. It lasts about 50 to 70 ns depended on the strength. Fitting an exponential decay on the data after the peak shows residuals of less than 1 mV, if we exclude the peaks. The rise in the integrator is mostly dependent on the synaptic strength, but also shows a slight dependency on V_{syntc}.

exponential decay to the data. We modify Equation (6.4) and obtain

$$V_{\text{integrator}} = \begin{cases} V_{\text{syn}} & \text{for } t \le t_0, \\ V_{\text{syn}} + \Delta V_{\text{integrator}} \cdot \exp\left(\frac{t-t_0}{\tau_{\text{syn}}}\right) & \text{for } t > t_{start}, \end{cases} \tag{7.4}$$

where t_0 is the time of the spike. We intentionally ignore the integration phase of the pulse here because it is short compared to the expected time-constants. Then $\Delta V_{\text{integrator}}$ can be extrapolated from the decay phase of the fit. This is useful, because the small overshoots at the begin and end of the integration phase are smoothed out be the model.

Fitting Equation (7.4) to the data shows that this simplified model matches well, but the small remaining deviation shows a systematic error, which is caused by the voltage dependency of the resistor. However, if we exclude the raising flank including the short peaks, the maximal residual of all fits is well below 1 mV. Plotting the fitted functions, would show no visible difference to the traces shown in Figure 7.4, except for the small overshoots. Therefore, we can conclude that this is no longer an issue to the functionality and call the decay as exponential.



A I_{syn} = 2.4 (——), 6.8 (·····) and 11 µA (- - - -)

B V_{syntc} = 0.27 (——), 0.45 (·····), 0.45 (- - - -), 0.45 (——) and 1.70 V (·····)

Figure 7.5 The plot shows strong synaptic pulses increasing the integrator voltage $\Delta V_{\text{integrator}}$ depended on the setting of V_{syntc} and I_{syn}. $\Delta V_{\text{integrator}}$ and τ_{syn} are obtained from fitting Equation (7.4). Two exemplary traces of the dataset are shown in Figure 7.4. The gray lines represent the values predicted by Equation (6.3). For smaller time-constants the pulse already decays significantly during the integration phase. Because of its finite gain, the integrator needs about 50 ns to integrate the pulse. This is not taken into account in Figure 7.4. This causes the effect to set in earlier than predicted by Equation (6.3). For larger time-constants $\Delta V_{\text{integrator}}$ converges to the predicted value. The dependency on I_{syn} is perfectly linear as described by Equation (6.3).

However the strongest pulses in our data increased $V_{\text{integrator}}$ only to 1.3 V, so for larger voltages in the integrator the deviation still might have a stronger effect. For the given data, we can indeed determine $\Delta V_{\text{integrator}}$ and τ_{syn} using the model.

The results for the rise of the integrator voltage $\Delta V_{\text{integrator}}$ are shown in Figure 7.5. The relation between $\Delta V_{\text{integrator}}$ and I_{syn} is linear as predicted by Equation (6.4). However, for the smallest possible time-constants we observe that $\Delta V_{\text{integrator}}$ becomes smaller than predicted. We compare this to the full model given by Equation (6.3), because the condition $t_{syn} << \tau_{\text{syn}}$ may not longer hold. This can explain the deviation only partly. We assume, that the finite integration time of 50 to 70 ns is the reason for the deviations.

We also compare the obtained time-constants with the resistance shown in Figure 7.2. For the ideal circuit these are given by $\tau_{\text{syn}} = R \cdot C$. If we take the integrator values obtained for $V_{\text{integrator}}$ = 1.01 V, we find that the effective time-constants obtained by the fit are about 5 to 20 % smaller than we would expect from the

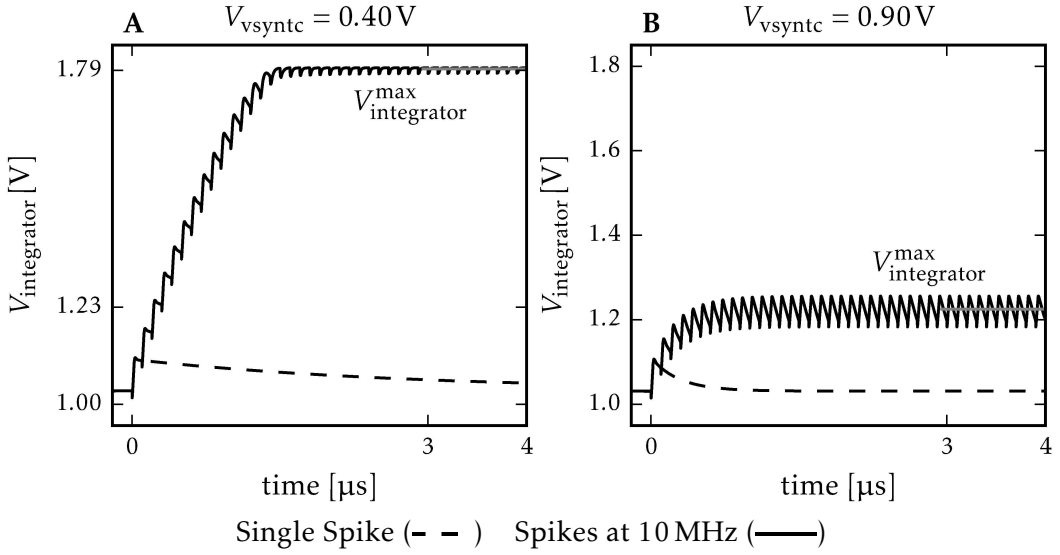


Figure 7.6 Simulations of the response in the integrator for synaptic current pulses of $t_{syn} = 10\,\text{ns}$ with $I_{\text{syn}} = 3.53\,\mu\text{A}$. The synaptic input is stimulated with a rate of 10 MHz. For comparison, a single spike is shown. We define the maximum voltage of the integrator $V^{\text{max}}_{\text{integrator}}$ as the averaged voltage after a settling period of 3 µs as defined by Equation (7.5). After this time the input reached a steady state for all tested parameters. The input can very well integrate a large number of spike events. The charging effect, we observed in revision 2, does not longer occur.

resistance value. This is also caused by the voltage dependency of the resistor. For the neuron functionality the effective time-constant is more important.

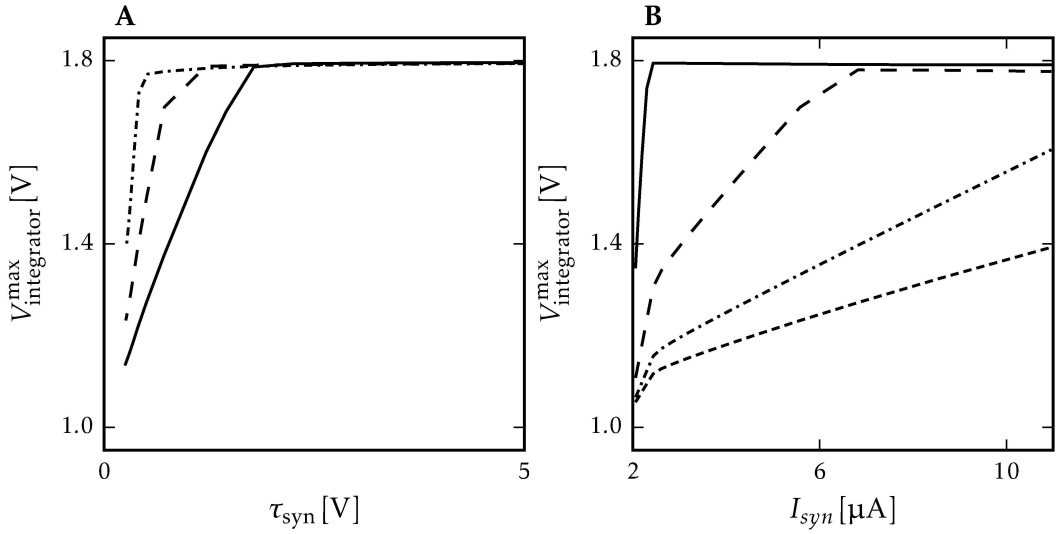
Multiple Spikes

When emulating neural networks, seldom only single spikes occur. We therefore repeat the simulation sending multiple spikes, but lower the input rate to 10 MHz, compared to Equation (7.5). This is still a rather high rate, a fifth of the maximum a single synapse can generate. The high rate helps to keep the simulation time short, because the integrator reaches equilibrium faster. As shown in Figure 7.6, the input integrates the spikes very well, until either a steady state or the maximum voltage of 1.8 V is reached.

Specifically, for this measurement, we define the maximum integrator voltage as $V^{\text{max}}_{\text{integrator}}$ as

$$V^{\text{max}}_{\text{integrator}} = \frac{1}{1\,\mu\text{s}} \int_{t_0=3\,\text{us}}^{t_0=4\,\text{us}} V_{\text{integrator}}(t)\,\text{d}t, \tag{7.5}$$

where the first spike arrives at $t = 0\,\mu\text{s}$. This definition leaves enough time for the



A $V_{syntc} = 0.40\,V$
$I_{syn} = 2.8$ (——), 5.6 (– –) and 11 µA (–·–·–)
B $I_{syn} = 2.5\,nA$
$V_{syntc} = 0.40$ (——), 0.60 (– –), 0.90 (–·–·–) and 1.70 V (– – –)

Figure 7.7 The plots show the maximum integrator voltage $V^{max}_{integrator}$ as defined in Figure 7.6. The results show that $V^{max}_{integrator}$ grows linear with τ_{syn} as well as I_{syn} as predicted by Equation (7.6). However, a quantitative comparison fails.

integrator to reach the equilibrium if weak pulses are used. The expected $V^{max}_{integrator}$ is given in Equation (6.6). But as we can now define a proper time-constant for the synaptic input, it is more suitable for us to replace R_{syntc}. We obtain

$$V^{max}_{integrator} = V_{syn} + \frac{f_{stim} \cdot \tau_{syn} \cdot I_{syn} \cdot t_{syn}}{C}. \tag{7.6}$$

In Figure 7.7 the $V^{max}_{integrator}$ obtained from the simulation is shown, where we obtained the synaptic time-constant, which label the axes, from the previous simulations of a single spikes. The data agrees with the behavior predicted by Equation (7.6). $V^{max}_{integrator}$ grows linear with both τ_{syn} and I_{syn}. For most settings, the integrator quickly reaches the supply voltage level of 1.8 V. For a quantitative comparison, this simulation needs to be repeated using various and especially lower stimulus rates.

As soon as the integrator voltage reaches 1.8 V, the input is completely saturated. Additional synaptic pulses will discharge the synapse line instead of being added to the integrator. Therefore, the synapse line needs to be recharged over R_{syntc} before the integrator can begin to decay. There are no indications of a negative effect on the neuron.

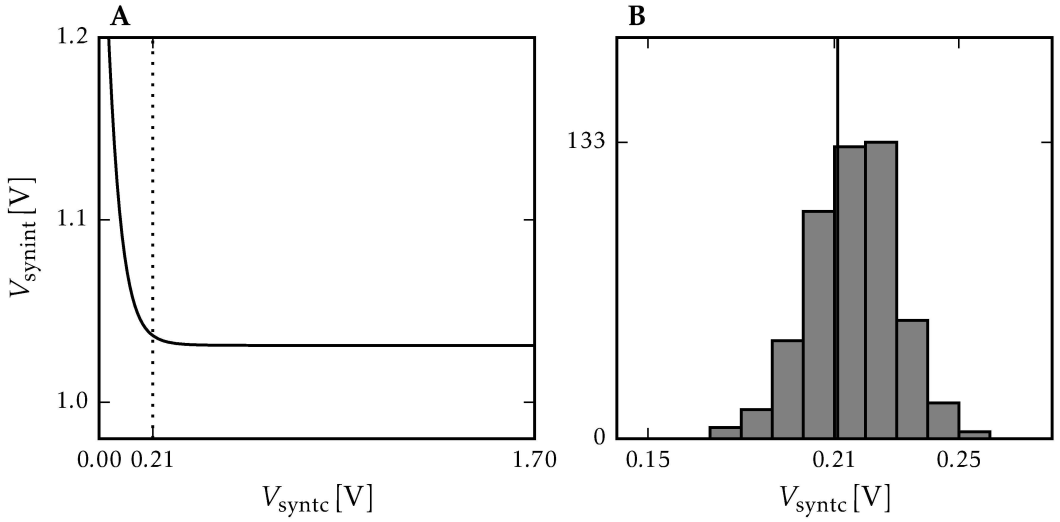


Figure 7.8 A Typical circuit simulation of the synaptic input with a sweep of V_{syntc} without any synaptic input. It can be observed that the voltage of the integrator is elevated below $V_{\text{syntc}} = 0.21 \pm 0.01\,\text{V}$. The vertical line marks a rise of 5 mV of $V_{\text{integrator}}$. **B** Monte Carlo simulation of the sweep ($N = 500$). Histogram of the V_{syntc} values causing a rise 5 mV in the integrator. Both simulations were conducted for $V_{\text{syn}} = 0.9\,\text{V}$.

7.1.3 Technical Limits for the Synaptic Time-Constants

The control parameter V_{syntc} of the synaptic time-constant is still subject to the effect that the integrator voltage rises, if the resistances become to large. We described the cause for this in Section 6.1.3. This sets an lower limit for V_{syntc} if the resting potential of the neuron shall not be affected by this. To estimate the onset of this effect, we conducted a Monte Carlo DC-simulation of the circuit and obtained a lower limit of $V_{\text{syntc}} = 0.21 \pm 0.01\,\text{V}$. The simulation is shown in Figure 7.8.

We see in Section 7.3, that the lower limit is actually larger than 0.21 V. This may have two reasons: Firstly, the neuron may already react to lower rises of the integrator voltage. Secondly, the simulation is imperfect. It just contains four synapses instead of all 220, which could influence the leakage current and the effect could be caused by the negative feedback loop with the OP, which is very sensitive to small deviations. We therefore expect this effect to vary stronger in the real hardware.

Further, the bias generator may be used to compensate for this effect, but this would require an individual setting for the bias generator dependent on the rise of the integrator voltage. Therefore, we so far did not follow up on this idea.

7.1.4 Summary

We conclude from the shown simulations that the changes of the synaptic input improve the functionality of the neuron. The new resistor provides a better controllable range of time-constants and its resistance is more stable against voltage differences compared to its counterpart in revision 2. The integration of spikes benefits from this and also from the improved scaling of the reference current for the synaptic event V_{gmax} and the larger capacity in the integrator. Replacing the conductance generating OTA_0 with a new resistor improves the dynamic range of the membrane significantly. Lastly and most importantly, the bias generator allows to compensate the DC-offset of OTA_1 and therefore to calibrate the minimum conductance for the synaptic input. This leads to a more stable neuron dynamic and allows smaller effective membrane time-constants.

7.2 Calibration of the Bias Generator

The first calibration task for the synaptic input is now always the bias generator. There is no reasonable default setting, because the DC-offset of OTA_1 is randomly distributed. To avoid problems with an uncalibrated input, we can disable the synaptic input by setting I_{conv} to zero and V_{convoff} to 1.8 V. The calibration of the bias generator works better if the membrane and the reversal potentials are already calibrated. Both synaptic inputs have to be calibrated independently. We therefore disable the input that is currently not calibrated, while we calibrate the other one.

The work on this calibration began already during the design phase. Without a calibration, the expected results are even in the best case as worse as for revision 2. Therefore it was crucial to demonstrate that the bias generator can be calibrated, before the new HICANN was produced. We could achieve this by integrating the neuron circuit simulation into our regular calibration framework, as described in Section 4.4.4. This had two advantages: Firstly, the bias generator was also thoroughly simulated in interaction with the complete neuron circuit. And secondly, the implementation of the calibration method in the calibration framework was already done when the first test-chips of HICANN revision 4 arrived in the lab. Then we completed the calibration routines using the Demonstrator Setups. Although this process was very successful, we will show here our latest measurements done on a HICANN revision 4.1 in a BrainScaleS System.

For the calibration we have to keep in mind that both the readout noise as well as the try-to-try variations add uncertainties to each data point, even if we only reconfigure V_{convoff}. To compensate for this, we first created a model linking the control parameter V_{convoff} to the shift of the membrane voltage introduced by the conductance and verified it by using simulations. This is shown in Section 7.2.1. Then we present in Section 7.2.2, how we used the model to create a calibration method and verified it on HICANN. Afterwards, we give a short outlook, how we can reduce the required steps for the calibration, and a short conclusion.

7.2.1 A Model for the Membrane Shift

The calibration of the bias generator has to use the membrane voltage, because this is the only voltage of the neuron we can readout on HICANN. Therefore, we will here create a model for the relation between the control parameter V_{convoff} and the membrane potential. We do not use any spike stimulus for this calibration, so that the neuron will be in the steady state for each measurement.

The basis for the calibration is that we can control the conductance of the synaptic input with the bias generator. It can shift the voltage at the terminals of OTA_1 arbitrarily to each other. This allows to freely control the output current I_{OTA_1}, which then is proportional to the conductance of the synaptic input. We have to keep in mind that a negative conductance is technically not possible. The point of interest here is the transition from a zero conductance to a non-zero conductance, because at that point the bias generator perfectly compensates the mismatch of OTA_1. We will need less measurements if the model describes the transition well enough, because then we can fit to the model and do not need to find the transition point directly.

For each setting of V_{convoff}, the membrane will settle in a steady state as described by Equation (2.7). Because we disable one of the synaptic inputs, we can remove one synaptic conductance term from Equation (2.7) and obtain

$$V_{\text{mem}} = \frac{E_{\text{syn}} \cdot \tilde{g} + E_l}{\tilde{g} + 1} \quad \text{with} \quad \tilde{g} = \frac{g_{\text{syn}}}{g_{\text{l}}}, \tag{7.7}$$

where E_{syn} is the inhibitory or excitatory reversal potential and g_{syn} the corresponding conductance of the active synaptic input. We see that we can obtain a direct relation between membrane voltage V_{mem} and synaptic conductance g_{syn} if we keep the leakage conductance g_{l} constant. Using Equation (7.7) we can relate V_{convoff} to a shift of the resting potential which we can measure.

To obtain a model for $\tilde{g}$, we simulate the synaptic input, using a typical DC-simulation, and obtain the current from the synaptic input I_{synx} onto the membrane and the resulting $\tilde{g}$ in dependency from V_{convoff}. We do this for two settings of the leakage conductance $I_{\text{gl}} = 150$ and $1500\,\text{nA}$ (scaling $1:3$). As we see in Figure 7.9, I_{OTA_1} and $\tilde{g}$ strongly depend on the chosen leakage conductance g_{l}. The relation between I_{synx} is linear, as long as the membrane is in the linear range of the leakage OTA. We can observe this well for the higher g_{l} setting, while I_{synx} quickly saturates for the lower g_{l} setting, because the membrane is pulled towards the reversal potential. Most importantly, we observe that the transition to the non-zero conductance is not very sharp. There is a V_{convoff} range of about $20\,\text{mV}$,where I_{synx} from the synaptic input slowly begins to rise before the relation between V_{convoff} and I_{gl} becomes linear. The ratio $\tilde{g}$ follows the current flow, as excepted. We notice here, that for high g_{l} the transition to a non-zero conductance is harder to detect, because here the ratio is still very close to zero in this range and will cause a non-measurable membrane shift.

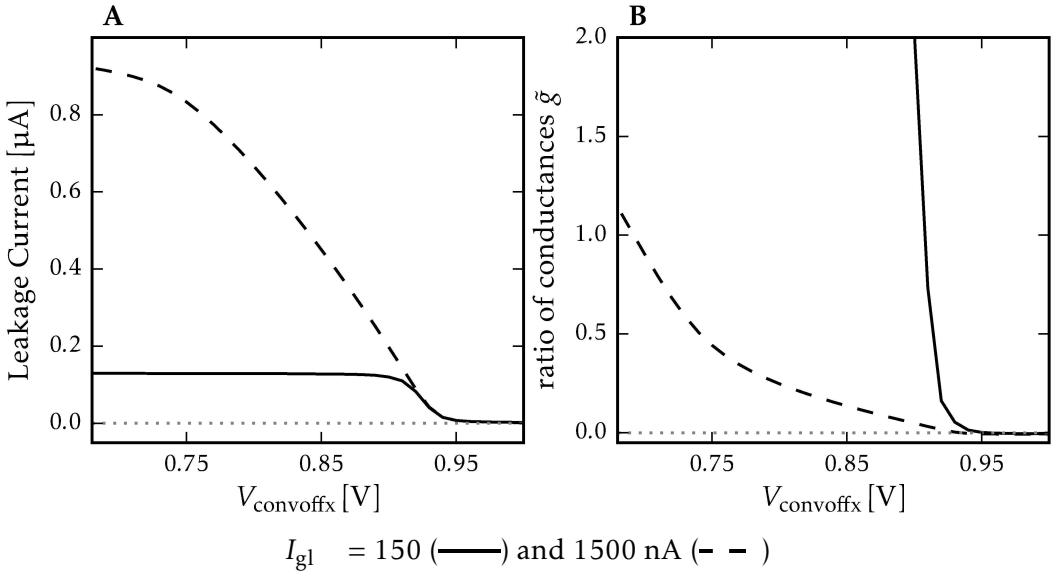


Figure 7.9 Typical DC-simulation using the simplified neuron simulation with synapses (see Section 4.4.3). Here the excitatory synaptic input is shown, but inhibitory input behaves qualitatively the same. The nominal zero point of OTA_1 is at 0.95 V taking its DC-offset into account (see Figure 6.1). **A** For each point the neuron is in a steady state. A current flows from the synaptic input over the membrane into the leak. It becomes larger as V_{convoffx} is getting smaller, because voltage difference at OTA_1 is then larger (see Figure 7.1). The current is proportional to g_x and causes the membrane to shift towards the reversal potential depended on I_{gl}. The limited linear input range of the leak OTA causes the current then to saturate, for low I_{gl} very quickly, for large I_{gl} more slowly. **B** The ratio of the resulting conductances $\tilde{g}$ is shown. For low I_{gl} it rises very fast, whereas for a high I_{gl} it rises quite slowly, so that the transition region becomes difficult to detect.

As long as the membrane potential is in the linear range of the leakage OTA we can use a linear function to describe $\tilde{g}$. Additionally, we need to take the smooth transition to the zero conductance value into account. We can here use the same softplus function as in Equation (7.3). However, this limits our method to membrane shifts not larger than 100 mV. We define our model for the ratio $\tilde{g}$ as

$$\tilde{g} = \frac{a}{\gamma} \ln\Big(1 + \exp\big(\gamma\,(b - V_{\text{convoff}})\big)\Big). \tag{7.8}$$

The parameter γ controls the curvature between the linear and the constant part of the transition, a the slope of the linear part and b the transition point. For large γ

the transition becomes more sharp, so that in the limit it becomes

$$\lim_{\gamma \to \inf} \tilde{g} = \begin{cases} a\,(b - V_{\text{convoff}}) & \text{for } V_{\text{convoff}} < b \text{ and} \\ 0 & \text{for } V_{\text{convoff}} \geq b. \end{cases} \tag{7.9}$$

By comparison with the simulations shown in Figure 7.9 we found that a value of $\gamma = 0.2$ describes the transition quite well for both I_{gl} settings.

We can improve the comparison between model and data by simplifying Equation (7.7). Because it is limited to membrane shifts smaller than 100 mV, $\tilde{g}$ is small. Therefore we can use a Taylor expansions around $\tilde{g} = 0$ and we obtain

$$V_{\text{mem}} \approx E_l + \left(E_{\text{syn}} - E_{\text{l}}\right) \cdot \tilde{g} + \mathcal{O}\left(\tilde{g}^2\right) \text{ for } \tilde{g} \ll 1. \tag{7.10}$$

By replacing $\tilde{g}$ with Equation (7.8) we can develop the final relation between V_{convoff} and the membrane potential. We obtain

$$V_{\text{mem}} = E_l + \left(E_{\text{syn}} - E_{\text{l}}\right) \frac{a}{\gamma} \ln\left(1 + \exp\left(\gamma\,(b - V_{\text{convoff}})\right)\right) \tag{7.11}$$

$$= E_l + \frac{\alpha}{\gamma} \ln\left(1 + \exp\left(\gamma\,(b - V_{\text{convoff}})\right)\right). \tag{7.12}$$

Because the parameters E_{syn} and a are not longer independent from each other, we combine them into the single parameter α.

To verify our model, we use the simplified neuron simulation as described in Section 4.4.3. We used a reversal potential of $E_{\text{l}} = 0.8$ V, a excitatory potential of $E_{\text{synx}} = 1.6$ V and two settings of $I_{\text{gl}} = 0.15$ and 1.5 µA. V_{convoff} was changed in steps of 10 mV from 0.0 to 1.8 V. Figure 7.10 shows two exemplary fits of Equation (7.12) to the typical simulations. We use a fixed $\gamma = 0.2$, as we do not have many data points in the transition region to obtain the parameter from a fit, especially for the higher I_{gl} value. The plot shows that Equation (7.12) approximates the data very well, independent from the chosen leakage conductance. This is of great advantage, because we can now use this approach without worrying about the need for a calibration for the leakage conductance. However, we need to exclude data points where the resting potential is more than 100 mV shifted from its set value.

We determine the transition point to a non-zero conductance by choosing the final V_{convoff} value from the fit, where

$$V_{\text{mem}}\,(V_{\text{convoff}}) = E_{\text{l}} + V_{\text{shift}} \tag{7.13}$$

is fulfilled. However, the chosen value for V_{shift} further influences the neuron behavior. For very low values, the conductance $\tilde{g}$ is still in the transition phase. This may cause weak spikes to be dampened. For larger V_{shift} the minimal conductance becomes larger, which leads to a faster membrane time-constant and requires a stronger leakage conductance I_{gl}.

Lastly, we compared the shift V_{shift} obtained from this fit to the actual shift

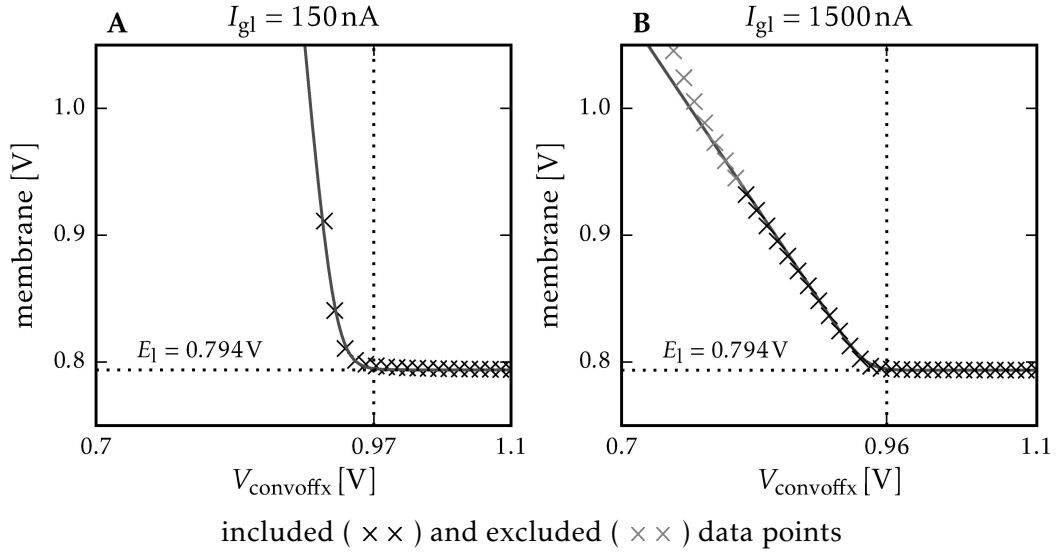


Figure 7.10 Typical DC-simulation using the simplified neuron simulation with synapses (see Section 4.4.3). $V_{convoff}$ was changed in steps of 10 mV and Equation (7.12) is fitted to the data. Values that are more than 0.10 V over the resting potential are removed (gray). The inhibitory input was deactivated by setting I_{convi} to 0 nA. The point where the fit indicates a 1 mV rise is marked. Even some of the removed points are still in the linear range, we need to choose this range conservative because it varies on the hardware.

obtained from the simulations in 500 Monte Carlo simulation runs. Figure 7.11 shows the results of a target shift of $V_{shift} = 1\,\text{mV}$. For a calibration using $I_{gl} = 150\,\text{nA}$, the actual shift is $4.2 \pm 0.8\,\text{mV}$, and for a calibration using $I_{gl} = 1500\,\text{nA}$, the actual shift is $1.5 \pm 0.3\,\text{mV}$. But if we evaluate the second shift at $I_{gl} = 150\,\text{nA}$ we obtain an actual shift of $6.3 \pm 1.6\,\text{mV}$, which is actually worse.

This deviation is caused by the imperfection of the transition phase and by the fixed chosen $\gamma = 0.2$. With the given model, we could improve the result by determining γ by the fit. However, we found that this made the fit unstable and would require more data points to map the transition phase. As we show in the evaluation of our model on HICANN, the decision for keeping γ as fixed parameter promises a better cost-value-ratio and still leads to satisfactory results.

In summary, we found in Equation (7.12) a sufficient model to relate the parameter $V_{convoff}$ to the change of the membrane potential. By incorporating the smooth onset into the model, it can be reliably used for a large range of I_{gl} settings. Hereby, we found a good balance between robustness and precision, especially as the result allows later corrections. Users can specify a membrane shift that would be tolerable, while not impairing weaker spike input too much.

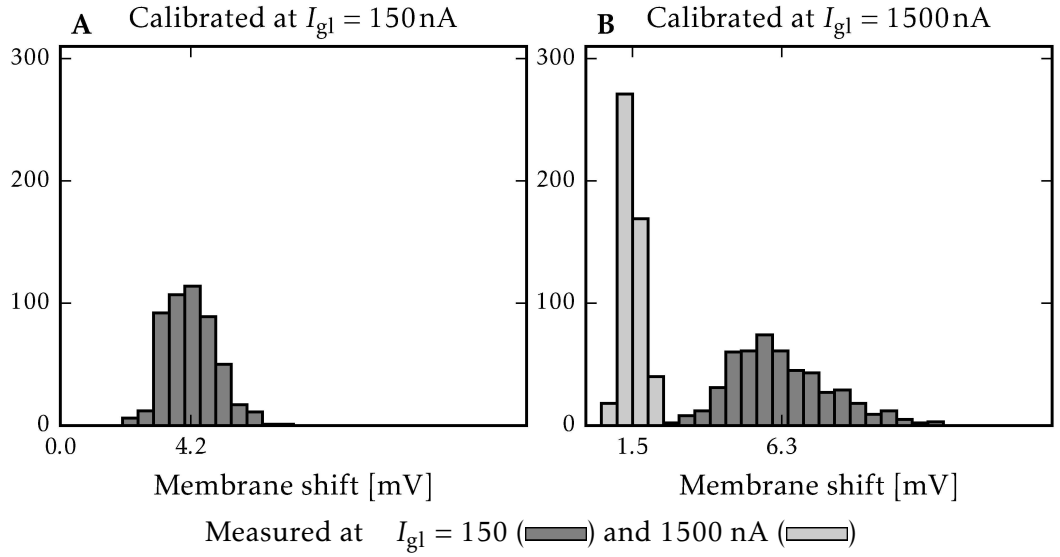


Figure 7.11 Monte Carlo simulation (N = 500) of the results shown in Figure 7.10. The neurons were calibrated using a V_{shift} = 1 mV at different I_{gl} of 150 and 1500 nA. The histograms show the actual membrane shift induced by the chosen $V_{convoffx}$ value. To make the results comparable, we obtained the shift for the calibration using I_{gl} = 1500 nA also at I_{gl} = 150 nA.

7.2.2 Calibrating the Bias Generator on Hardware

With the theoretical background we can apply the calibration on HICANN. The data shown here was for $V_{convoff}$ with 61 measurements from 0 to 1.8 V in step of 30 mV. The complete parameters are listed in Appendix A.5.3. Because the simulation showed similar results for different I_{gl} settings, we repeated this measurement using three different settings of I_{gl} = 150, 300 and 1500 µA using 1:3 scaling to find the most suitable setting.

Figure 7.12 shows the result of the measurement steps for an exemplary neuron with Equation (7.12) fitted onto the data. Data points with a distance larger than 100 mV to the resting potential were discarded. We do not recommend a higher limit, because the linear range of some OTAs is smaller than usual. For weak leakage conductances, the relevant $V_{convoff}$ range covers only about 5 data-points or 150 mV. For lower $V_{convoff}$, the conductance of the synaptic input becomes too large and the membrane is pulled outside the linear range of the leakage OTA. For the excitatory input the neurons will begin to spike. We detect this by evaluation of the peak-to-peak height of the trace, which then become larger because threshold and reset potential are set far apart. For a stronger leakage, the conductance grows linear over a range of at least 200 mV beginning at the onset. This matches the prediction well which we gathered from the model. But we have to keep in mind that the leakage conductance g_l is rather strong. This makes the weak onset of the synaptic conductance g_x harder to detect, because the membrane shift is proportional to the

ration $\tilde{g}$ of both. The medium leakage value shows that there is a smooth transition between both extremes.

For testing the performance of the different steps we made a second measurement. Therefore, we selected for each neuron 12 V_{convoff} settings for each input using a V_{shift} of 1, 2, 3 and 4 mV for each of the I_{gl} settings. Afterwards, we measured the distribution of the membrane potentials for various settings of the leakage conductance using I_{gl} = 0, 0.12, 0.24 and 0.49 µA (scaling 1:3). Figure 7.13 shows an exemplary result and the standard deviations for all settings. The setting I_{gl} = 0 µA is a test case. It shows that without the correcting leakage conductance even the smallest conductance shifts the membrane. The wide distribution shows that there are no unexpectedly dominating neuron terms. For the small leakage of I_{gl} = 0.12 µA, we observed significant differences between the calibrations. It clearly convinced us, that a small leakage conductance is preferable for calibration.

Given this result, we decided to conduct the calibration using I_{gl} = 150 µA and recommend to select a V_{shift} of 1 mV. The selected offset correction V_{convoff} is shown in Figure 7.14. The selected values are located in the middle of the possible parameter range.

We now want to estimate the error induced by the bias generator itself, because we do not have full models supporting Monte Carlo simulations for the transistors used in the generator. As reference we use the variation of the OTA offset σ_{OTA1} = 23 mV, obtained in Section 6.1. Using the ideal scaling factor of 0.28, we obtain a scaled $\tilde{\sigma}_{\text{OTA1}}$ = 82 mV. The standard deviation of the V_{convoff} distribution is σ_{convoff} = 111 and 118 mV only a little larger. This gives us the deviation of the bias as

$$\sigma_{\text{bias}} = \sqrt{\sigma^2_{\text{convoff}} - \tilde{\sigma}^2_{\text{OTA1}}} = 76\,\text{mV}. \tag{7.14}$$

Note that the comparison assumes that the distribution is normal and ignores the error of the scaling factor. However, as estimated we see that offset and bias generator contribute equally into the variation. Lastly, we can say that the scaling factor is well chosen, because it provides a sufficient precision and makes good use of the available parameter range.

7.2.3 Outlook on Optimizing the Required Steps

We so far require a fairly large number of steps to cover the complete parameter range with sufficiently small steps. We can improve this by implementing a two step calibration. A first coarser step is used to determine the approximate location of the transition phase to a zero conductance and then the transition range is scanned using finer steps. We can reduce the number of steps required in the coarse scan if we use the model developed in Section 7.2.1. However, the approximation used for the finer measurement is no longer suitable. We insert Equation (7.8) directly into

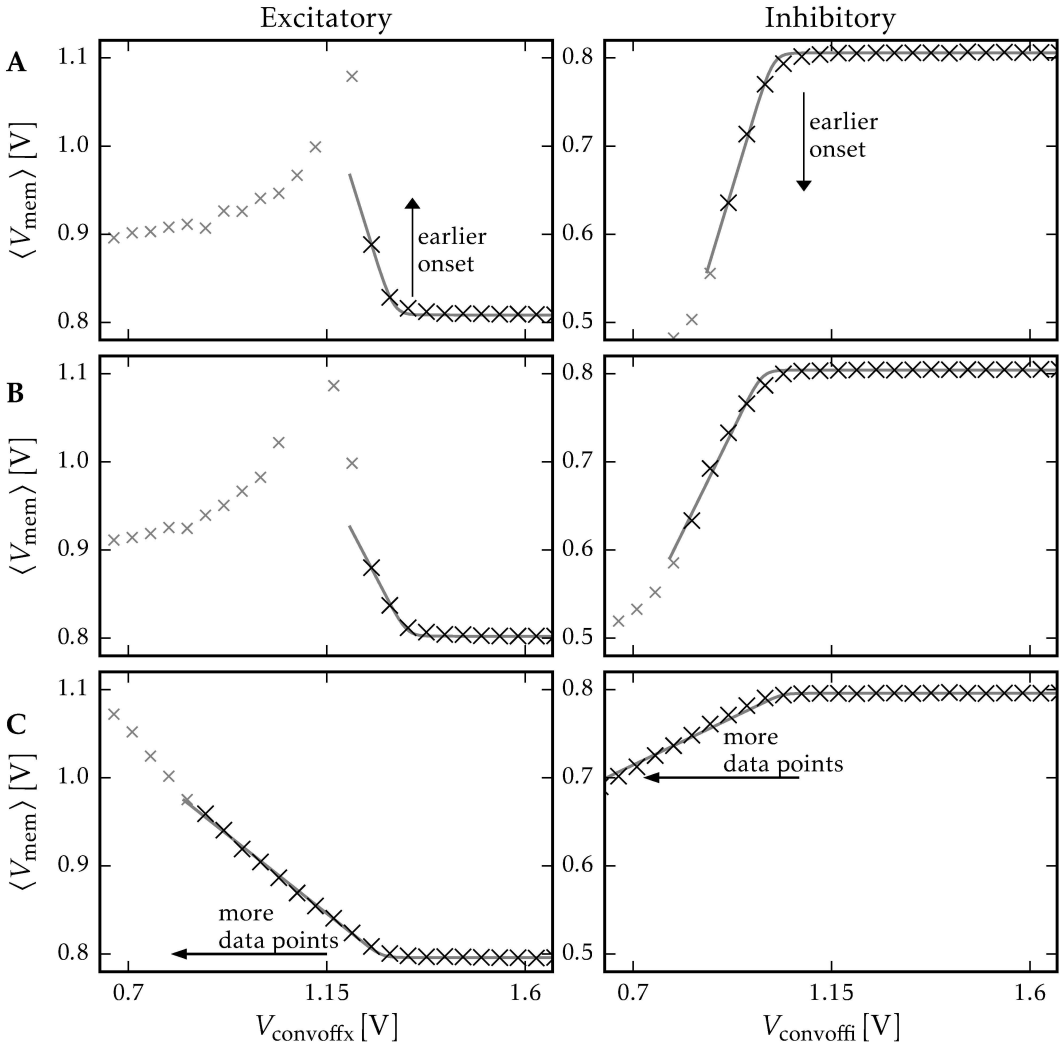
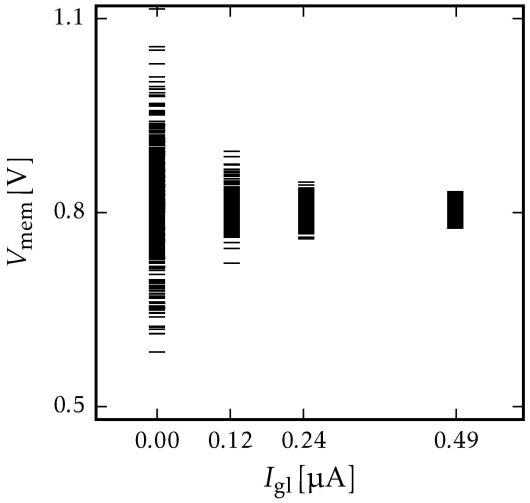


Figure 7.12 Measurement for V_{convoff} with different leakage conductances . For this measurement 61 steps of V_{convoff} from 0.0 to 1.8 V were done, where only the V_{convoff} floating gate parameter is reconfigured. The input that is not calibrated is deactivated. The plots show the mean membrane potentials recorded for each step and the fit of transformation function Equation (7.12) for neuron 413. The gray points are ignored for the fit because the membrane potential is more than 0.1 V away from the resting potential or the neuron began to spike. For the first criteria, we take for the excitatory inputs the lowest measured point as reference and for inhibitory inputs the highest. The second criteria can be detected, because we set spike threshold and reset potential far apart and use the peak-to-peak height of the trace to filter spiking neuron. This causes the mean to fall again for low V_{convoffx} values. The bended region of the measured points is not always perfectly matched by the recorded data points. But determining the curvature with the fit is even more unreliable because of the few data points available in this region.



Standard deviations at $I_{\text{gl}} = 0.12\,\mu\text{A}$ for different calibration parameters.

cal. I_{gl} [µA]	0.15	0.30	1.50
offset [mV]	σ [mV]	σ [mV]	σ [mV]
1	12	17	26
2	15	21	32
3	14	22	32
4	16	25	35

Figure 7.13 Measurement to validate the V_{convoff} calibrations. The shown calibration was done at $I_{\text{gl}} = 0.3\,\mu\text{A}$ with a target offset of 3 mV. To test the quality of the calibration, we determined the membrane potential for each neuron at different settings of the leakage bias current I_{gl}. The resulting distributions over all neurons are plotted. The resulting standard deviation of the distribution is annotated. The table presents the standard deviations obtained for $I_{\text{gl}} = 0.12\,\mu\text{A}$ for various other calibrations settings. The result shows that for higher I_{gl} settings during calibration the distribution of membrane potentials widens indicating a less precise calibration.

Equation (7.7) and obtain

$$V_{\text{mem}} = \frac{E_{\text{syn}} \cdot \tilde{g} + E_l}{\tilde{g} + 1} \quad \text{with} \quad \tilde{g} = \frac{a}{\gamma} \ln\Big(1 + \exp\big(\gamma\,(b - V_{\text{convoff}})\big)\Big), \tag{7.15}$$

where we now use a fixed $\gamma = 0.5$ for the fit. This model still requires that the membrane is in the linear range of the leakage OTA. Therefore, we adapt the parameters to $E_{\text{syni}} = 0.5\,\text{V}$, $E_{\text{synx}} = 1.1\,\text{V}$ and $I_{\text{gl}} = 2.5\,\mu\text{A}$ (scaling 1:3).

Figure 7.15 shows the result on an exemplary neuron using 10 steps from 0.0 to 1.8 V for V_{convoff}. The plot shows that the model still agrees well with the data. In particular if we keep in mind that we do not need a very high precision because the goal of the measurement is it to obtain an approximation for a finer measurement.

We compare the approximation with the final calibration results obtained in the previous section using 61 steps. As shown in Figure 7.16, all found target values lie within 0.3 V below the value found in the measurements. Because the value we found lies for sure below the reference from the full measurement, we could now conduct a second measurement consistent of 11 steps from b to $b + 0.3\,\text{V}$, using steps of 30 mV like in the previous section. We were not able to integrate this into the

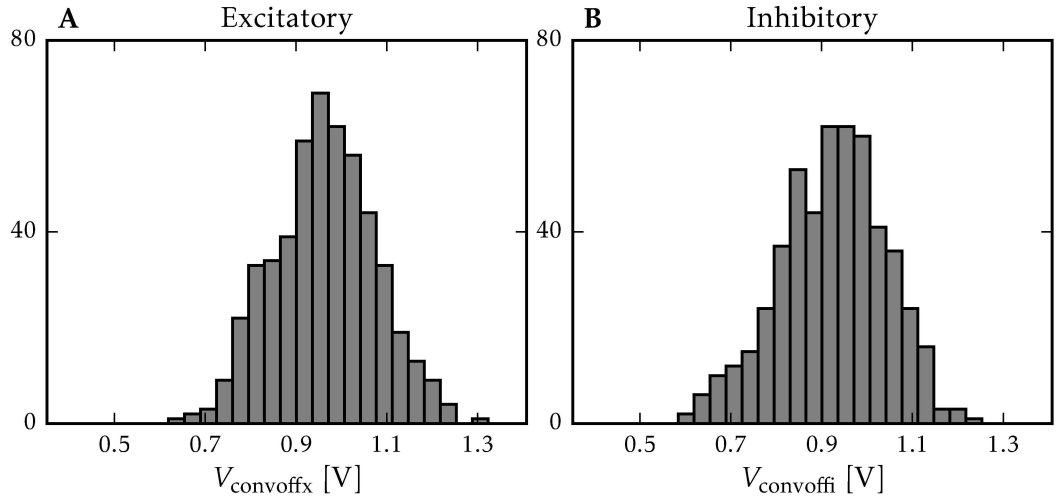


Figure 7.14 Result of the V_{convoff} calibration. The calibration was done at $I_{\text{gl}} = 0.150\,\mu\text{A}$ with a target offset of 1 mV. The width distribution is $\sigma_x = 111\,\text{mV}$ and $\sigma_i = 118\,\text{mV}$. Because we do not know the effect of mismatch on the level shifter we cannot directly compare this to the known offset distribution of the OTA. The chosen V_{convoffi} and V_{convoffx} values have a Pearson correlation coefficient of 0.02.

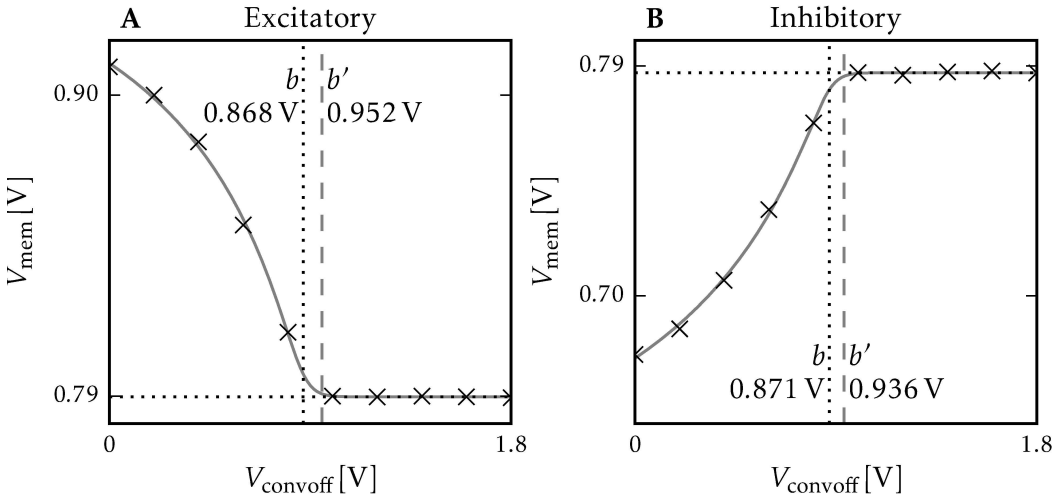
calibration software yet but we expect that this would lead to results with the same precision as the full measurement by using only a third of the steps.

7.2.4 Summary

The bias generator can be successfully used to compensate the DC-offset of OTA_1. The calibration requires us to scan almost the complete parameter range in fine steps, because the parameter is randomly distributed. A two phase scheme has been proposed to reduce the number of steps significantly. We found that the calibration leads to the best results if small I_{gl} values are used, because then the onset of the conductance is detected best. The final deviation from the target resting potentials is not much larger as directly after the calibration of the resting potentials shown in Section 5.2.

7.3 Calibration of the Synaptic Input

Based on the calibration of the bias offset we can continue with calibration of the parameters of the neuron. We first calibrate the reversal potentials and afterwards the synaptic time-constants. We then show that the calibration method for the synaptic time-constant can also be applied to the membrane time-constant. We



A $I_{\mathrm{gl}} = 2.5\,\mathrm{uA}$ and $E_{\mathrm{synx}} = 1.1\,\mathrm{V}$ **B** $I_{\mathrm{gl}} = 2.5\,\mathrm{uA}$ and $E_{\mathrm{synx}} = 0.5\,\mathrm{V}$

Figure 7.15 To reduce the number of steps required we propose running first a coarse calibration with less steps, and then only scan the relevant range with finer steps. Here a measurement with a coarse stepping of V_{convoff} in 10 equal distant steps from 0 to 1.8 V is shown. We reduced the distance to the reversal potential to keep the membrane in the linear range of the leakage OTA. This allows to fit Equation (7.7) to the data to take advantage of the whole measurement range. We show the fit parameter b obtained in this measurement and the one obtained in reference measurement b' using the full 61 steps measurement presented in Section 7.2.2.

shortly discuss the current speed of the calibration and then evaluate the calibration. We close with a short summary.

7.3.1 Calibration of the Reversal Potentials

The calibration of the inhibitory reversal potential remained almost unchanged compared to revision 2. However, we can now use the bias generator to create the required synaptic conductance to pull the neuron towards the reversal potentials instead of using spike input for this. The result of the calibration is shown in Figure 7.17, the parameters are listed in Appendix A.5.2. The calibration does not improve the spread of the distribution, because the resistor does not change the voltage provided by the floating gate cell. The spread is therefore in both cases caused by the trial-to-trial and the mismatch variations of the floating gate cell. The offset between calibrated and uncalibrated is in this case mainly caused by the deviation f_c from the perfect linear translation of the digital parameter value, as defined in Section 4.1.2

Unfortunately, a direct calibration of the excitatory reversal potential for values

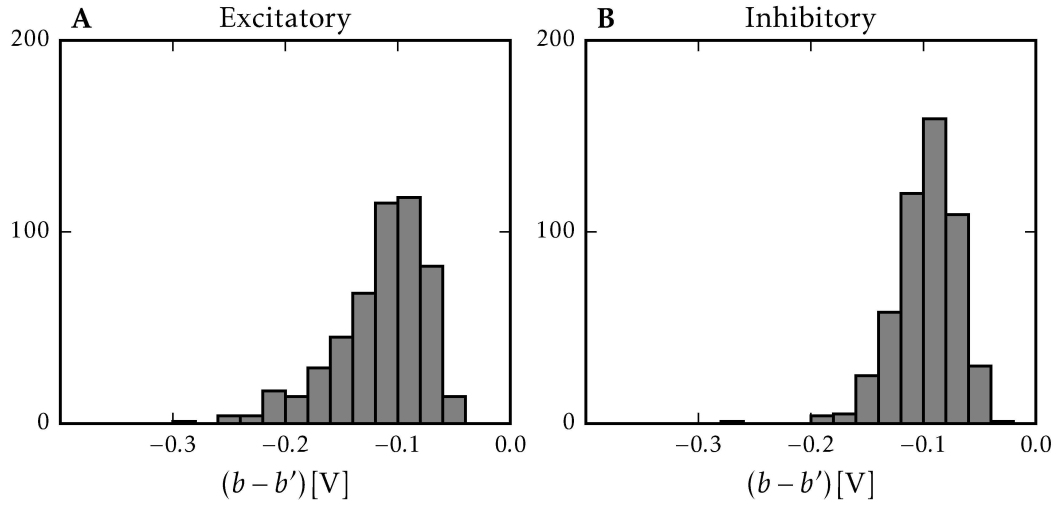


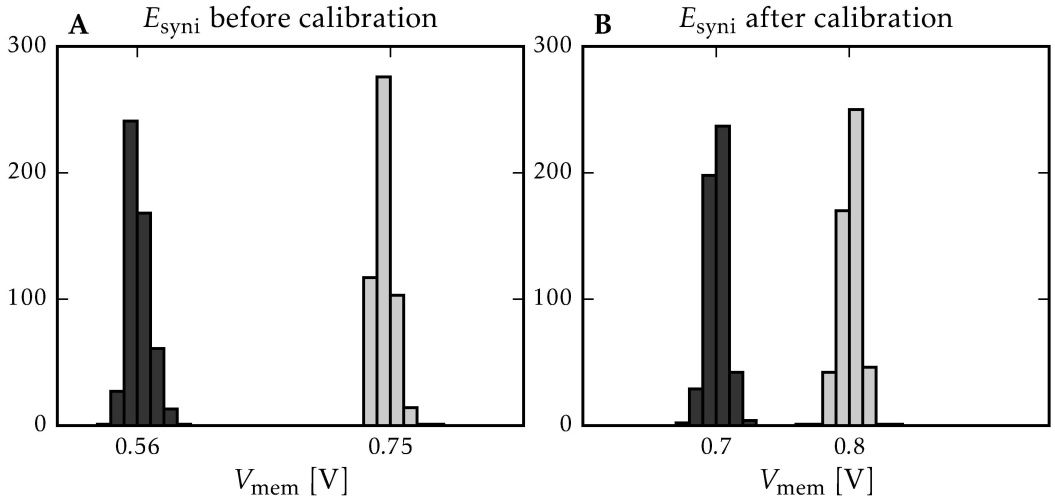
Figure 7.16 Comparison of a reduced V_{convoff} calibration measurement with the full 61 steps sweep and $I_{\text{gl}} = 150\,\text{nA}$ presented in Section 7.2.2. The reduced calibration uses 10 equidistant steps from 0 to 1.8 V and fitted against Equation (7.7). We compare the fit parameter b from the reduced calibration with the one from the reference measurement b'. The histogram shows that we could reduce the calibration range for each neuron to a range of 0.3 V including a safety margin.

larger than 1.2 V is not possible, because the membrane voltage is limited to values below that point, as discussed in Section 4.2.3. An extrapolation from the lower range would induce a large error, caused by the non-linear course of the conductance. However, we can still test the reversal potential for values below 1.2 V in this case we find the same distribution as for the inhibitory input of about 8 mV for all measured points. This means that the spread for an uncalibrated reversal potential is not larger than for a calibrated, like we would expect it.

We further intend to set the excitatory reversal potential quite high to about 1.4 to 1.6 V, maximizing the obtained conductance. In this range, the non-linearity of the conductance will dominate over the deviation of the reversal potential and would render a regular calibration useless. For the practical use, the uncalibrated excitatory potential shows so far only little downsides.

7.3.2 Calibration of the Synaptic Time-Constants

The basic principle for the calibration of the synaptic time-constant remains unchanged compared to Section 6.2.3. However, we need to adapt the process to the changes in the synaptic input and develop a model for the dependency of the synaptic time-constant τ_{syn} on the control parameter V_{syntc}. We evaluate the calibration for both synaptic inputs. The parameters of the measurement shown are noted



	A before calibration		**B** after calibration	
	target[V]	$\mu \pm \sigma$[V]	target[V]	$\mu \pm \sigma$[V]
	0.6	0.561 ± 0.008	0.7	0.701 ± 0.007
	0.8	0.746 ± 0.007	0.8	0.801 ± 0.008

Figure 7.17 Distribution of the inhibitory reversal potentials before and after calibration on HICANN revision 4.1. Because the resistor is only a passive element, the deviation of the evaluation measurement does not change. It directly reflects the trial-to-trial variation of the floating gate parameter. Bin-size: 10 mV.

in Appendix A.5.4. The result of the calibration for a single neuron is shown in Figure 7.18.

Selecting Post-Synaptic-Potentials

As discussed before, not every measured PSP is suitable to determine the synaptic time-constant. We therefore reuse the methods shown in Section 6.2.1 to evaluate each one. We keep the thresholds for the signal criterion and for the tolerance for a shift of the resting potential E_{l} as before. Nevertheless, we need to adjust the limit for the reduced χ^2_{red} criteria slightly to accept a higher limit of $\chi^2_{\mathrm{red}} < 5$, because the position of the highest peak of the χ^2_{red} distribution varies between 0.7 and 2.0 over various measurements. However, the distribution obtained is still similar to the one shown in Figure 4.5, especially the fact that most of the fits either lead to an χ^2_{red} near the highest mode or show a very large deviation instead. Therefore, this change has little effect on the final results. We reject all data-points that do not fulfill all three criteria. If less than six of data points remain, we mark the input as defect.

Figure 7.19 shows the number of data points discarded by each of the criteria. Also almost all synaptic inputs show a good response to the stimulus. We find in

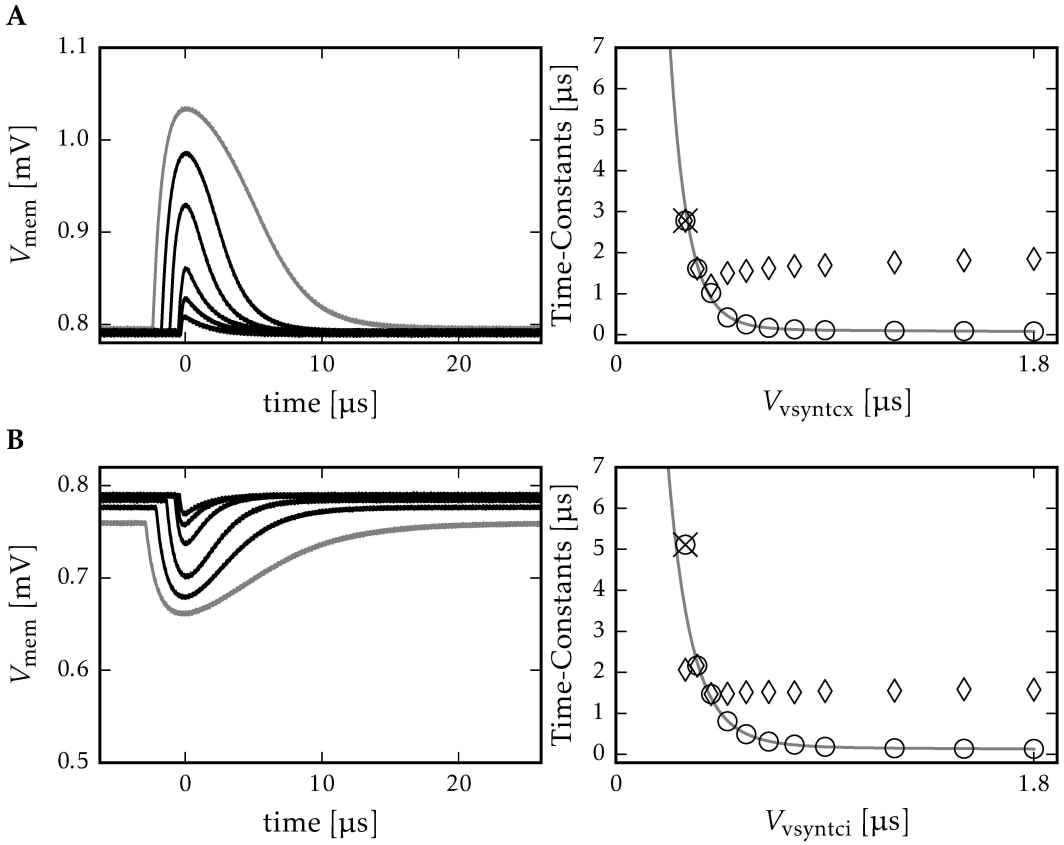


Figure 7.18 Exemplary traces of an **A** excitatory and **B** inhibitory V_{syntc} calibration. The traces are aligned at their extrema and rejected traces are gray. The resulting time-constants and the fit of the transformation function are shown on the right side. We can see that the time-constants are mostly clearly distinguishable. However, at the point where both are almost equal, the fit does not converge very well and the result becomes slightly distorted. The time-constants follow the expected behavior and we can fit Equation (7.16).

the excitatory input only 11 neurons that have a weak response. This was observed before [Kungle 2016], but is not yet understood. Most of the PSPs have a good shape, so that the fit has a $\chi^2_{red} < 5$. However, this only holds true for V_{syntc} below about 0.4 V on the excitatory and below about 0.35 V in the inhibitory input. For lower settings, the rejected PSPs show either large PSPs with heights larger than 150 mV or the resting potential is shifted indicating a rise in the integrator voltage $V_{integrator}$ as described in Section 7.1.3. The difference between excitatory and inhibitory input is caused by the differently chosen reversal potentials and their different characteristics.

Opposed to revision 2, many of the PSPs rejected by the χ^2_{red} criteria still would

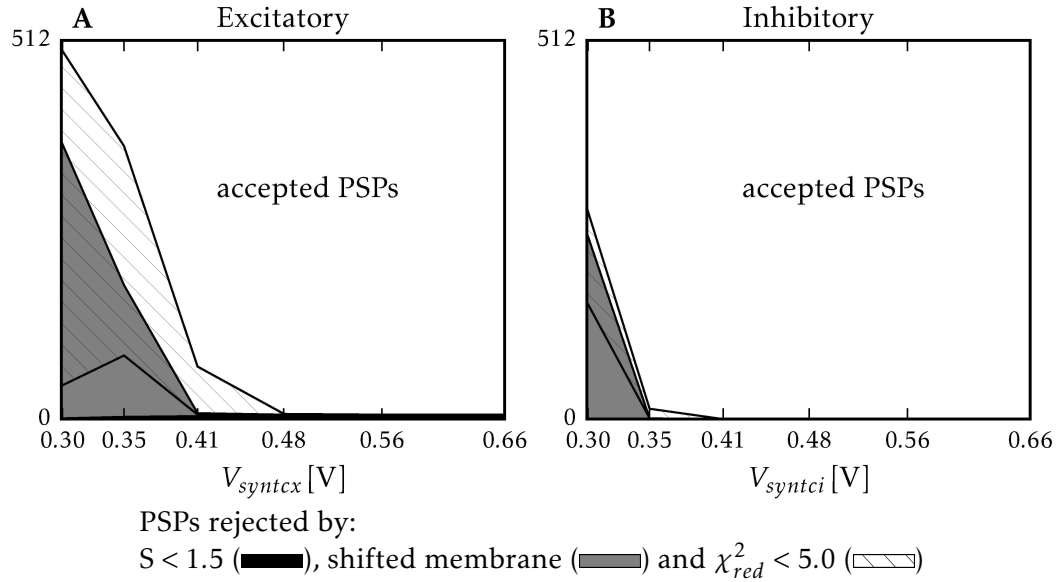


Figure 7.19 Selection of valid data points for the synaptic time-constant calibration. We restricted the plot to the lower V_{syntc} settings. For the values above 0.66 V, for the exitatory input the number of PSPs rejected by the signal-to-noise criteria increase to 11 at 1.8 V. The inhibitory inputs show no additional defects.

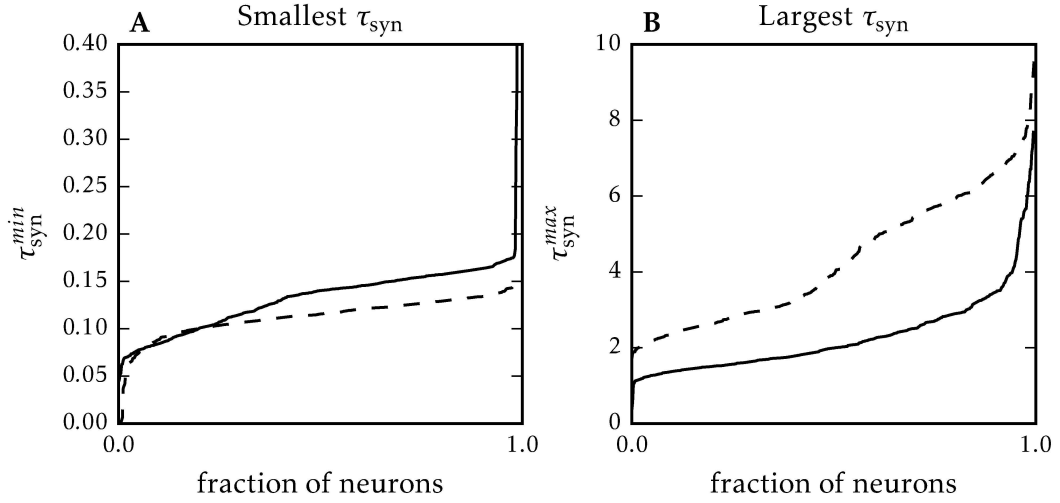
hold an inspection by eye. While saturation effects causing plateaus do no longer occur, like in revision 2, the time-constants obtained by the fit are no longer reliable. We notice that most of this PSPs would also been discarded because the resting potential of the neuron was already shifted. For the given data-set only five inputs had not enough valid data points.

Fitting the Time-Constant

Before we can fit the model for the time-constants we need to sort them. As discussed in Section 2.1.2, τ_1 and τ_2 obtained from the fit need to be matched to the membrane time-constant τ_m and the synaptic time-constant τ_{syn}. However, as shown in Figure 7.2, the synaptic time-constant becomes at some point larger than the membrane time-constant, so that simple sorting is no longer sufficient. But we still can well distinguish the time-constants for higher V_{syntc} settings. Here we can safely assign the larger one to τ_m. Therefore, we take the average of τ_m for $V_{\text{syntc}} > 0.7$ V and assign for the remaining data-points τ_1 and τ_2 based on the distance to this reference value. We classify the closer value as τ_m and the other as τ_{syn}.

Lastly, we can match V_{syntc} with the obtained time-constants. We fit the relation

$$\log \tau_{\text{syn}} = \frac{a}{c} \cdot \log\left(1 + \exp\left(c \cdot (V_{\text{syntc}} - b)\right)\right) + d \cdot V_{\text{syntc}} + \tau^0_{\text{syn}}, \tag{7.16}$$



	τ_{syn}^{min} [µs]		τ_{syn}^{max} [µs]	
Synaptic Input	mean	99.7 % within	mean	99.7 % within
Excitatory (——)	0.14 ± 0.13	0.07 to 0.18	2.28 ± 1.05	1.14 to 6.00
inhibitor (– –)	0.11 ± 0.02	0.04 to 0.15	4.30 ± 1.70	1.99 to 7.99

Figure 7.20 Minimal and maximal time-constants for each neuron. For the maximal time-constant the highest measured values, that was accepted by the criteria shown in Section 7.3, is shown. The table denotes the mean time-constant and the standard deviation as well as the range covered by 99.7 % of the neurons. If a certain membrane shift and uncertainties in the time-constant are accepted most neurons can reach time-constants above 4 µs. The difference between inhibitory and excitatory input is caused by the asymmetric conductances of the inputs. Because the excitatory conductance is stronger, the neuron is more susceptible to a rise of the integrator voltage or badly shaped PSP.

on the logarithm of the data, which we obtained by using the relation $\tau_{syn} = R_{syntc} \cdot C$ on Equation (7.3). The fit is robust, if the initial parameters are well chosen.

Reachable Time-Constants

The resulting minimal and maximal time-constants are shown in Figure 7.20. It is partly possible to use a larger time-constant as long as the resting potential of the neuron does not get shifted. However as for these time-constants the χ^2_{red} criteria failed, they may lead to unreliable results and unexpected behavior.

7.3.3 Calibration of the Membrane Time-Constants

We can also use the PSP based measurement to calibrate the membrane time-constant. This is a straight forward modification of the calibration for the synaptic time-constant. We will shortly present the results here, because this is an verification of the method we shown. However other methods for this already exist [Schwartz 2013; Schmidt 2014]. We use for the calibration 10 steps for I_{gl}. The parameters of the measurement shown are noted in Appendix A.5.4.

In contrast to the synaptic time-constant calibration the amplitude of PSPs is more homogeneous across the whole parameter range. This agrees with Equation (2.8), where τ_m occurs in the nominator as well as in the denominator. Therefore, all datapoints show very good PSPs, and none is rejected by the criteria shown before. The leakage conductance g_{l} is proportional to the square root of the bias current [Millner 2012]. In addition we need a linear correction to take the linear error of the floating gate into account. Therefore, we can describe the membrane time-constant τ_{m} by

$$\tau_{\mathrm{m}} = \frac{a}{\sqrt{I_{\mathrm{gl}} - c}} + \frac{b}{I_{\mathrm{gl}}}, \tag{7.17}$$

where we fit the parameter a, b and c. We show some exemplary traces and the fit in Figure 7.21.

The reachable membrane time-constants are shown in Figure 7.21. For a an I_{gl} scaling of $1 : 3$ we have a lower limit of $0.47 \pm 0.09\,\mu\mathrm{s}$ and an upper limit of $3.3 \pm 1.8\,\mu\mathrm{s}$. The results of Schmidt [2014] show larger values for both the minimal as well as the maximal values. This is caused by the difference in the calibration method: in his work the current stimulus of the neuron is used, which adds an additional capacity to the membrane. So far the results agree with each other, but a thorough comparison of the methods remains to be done. Lastly, the calibration of the membrane time-constant using PSPs confirms that the method works reliable.

7.3.4 Evaluation of the Calibration

We evaluate the calibrations for membrane and synaptic time-constants by measuring three time-constant values for each of the parameter. To estimate the trial-to-trial error we repeated those $N = 30$ times. The parameters of the measurement are listed in Appendix A.5.6.

First, we notice that the result shows a systematic error: the obtained time-constants are generally larger than expected. But this occurs only in the exponentially growing regions of the transformation functions. This gives us an explanation for this behavior: The normally distributed trial-to-trial gets skewed due to the transformation in the circuits. Therefore, the measured time-constant are no longer normal distributed, but skewed, in the exponentially growing region. It is more likely to obtain a overly large time-constant in a single measurement than a smaller one. This affects the fit and the obtained transformation functions show the observed

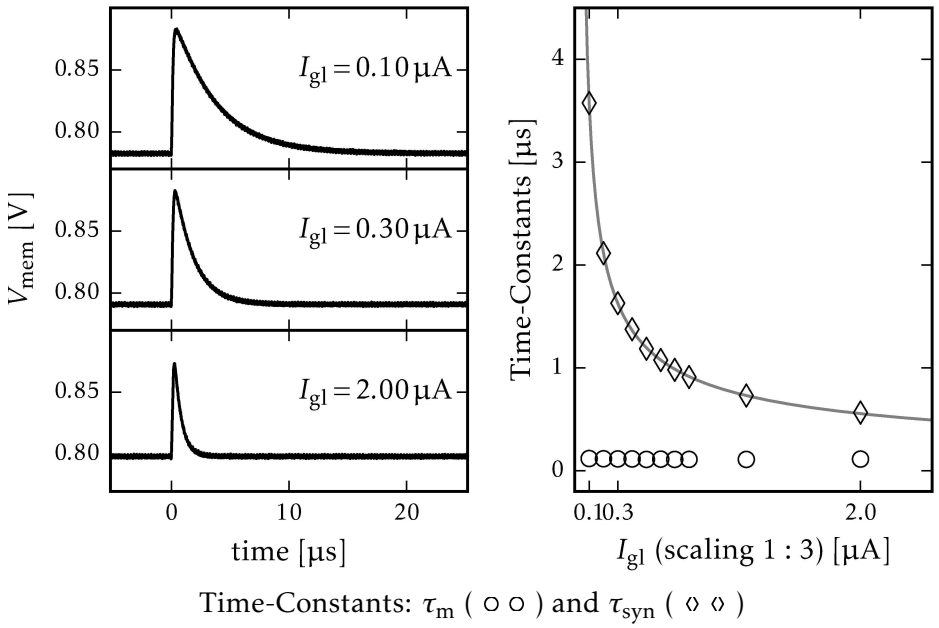


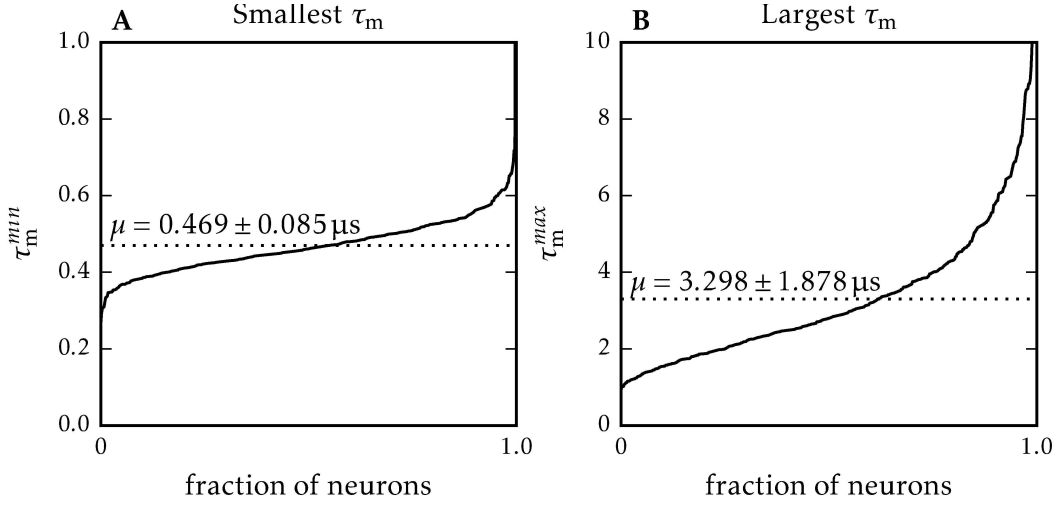
Figure 7.21 Exemplary traces of a membrane time-constant calibration. The traces are aligned at their onset. The resulting time-constants and the fit is shown on the right side. We can see that the time-constants are clearly distinguishable. They follow the expected behavior and we can fit Equation (7.17). The data-points match the model very well, all residuals are within a 0.01 µs range around the model.

systematic deviation.

For the synaptic time-constant, the systematic error is about 20 % for a target value of $\tau_{syn} = 1\,\mu s$. For the membrane time-constant, we find a systematic error of 40 % for a target value of $\tau_m = 2\,\mu s$. The difference between both is caused by the different slopes of the transformation functions at this points. In contrast to the trial-to-trial variation, this effect can be compensated by the user, either manually or by using learning algorithms in the network.

The comparison with the uncalibrated data shows that we can indeed reduce the spread of the measured distributions. For synaptic time-constants of $\tau_{syn} \leq 0.5\,\mu s$ we reach the same level as given by the trial-to-trial error σ_t, that we obtained from the repeated measurement. For the larger time-constants, which lie in the exponentially growing region of the transformation function, we still reduce the spread of the measured distribution, but no longer reach the precision of the trial-to-trial error σ_t. The final results for synaptic time-constant are presented in Figures 7.23 and 7.24

For the membrane time-constants τ_m, we get for all tested time-constants the spread of the distribution of the measured time-constant for all three measurements equally close to the trial-to-trial error. While the total final precision is not as good as for the synaptic time-constants, the gain in precision is much larger, in total a



	τ_m^{min} [µs]		τ_m^{max} [µs]	
	mean	99.7 % within	mean	99.7 % within
——	0.47 ± 0.09	0.34 to 0.65	3.30 ± 1.88	1.14 to 8.95

Figure 7.22 Minimal membrane time-constant τ_m^{min} and the maximal τ_m^{max}. The table denotes the mean time-constant and the standard deviation as well as the range covered by 99.7 % of the neurons. The minmal and maximal time-constants have a Pearson correlation coefficent of 0.50.

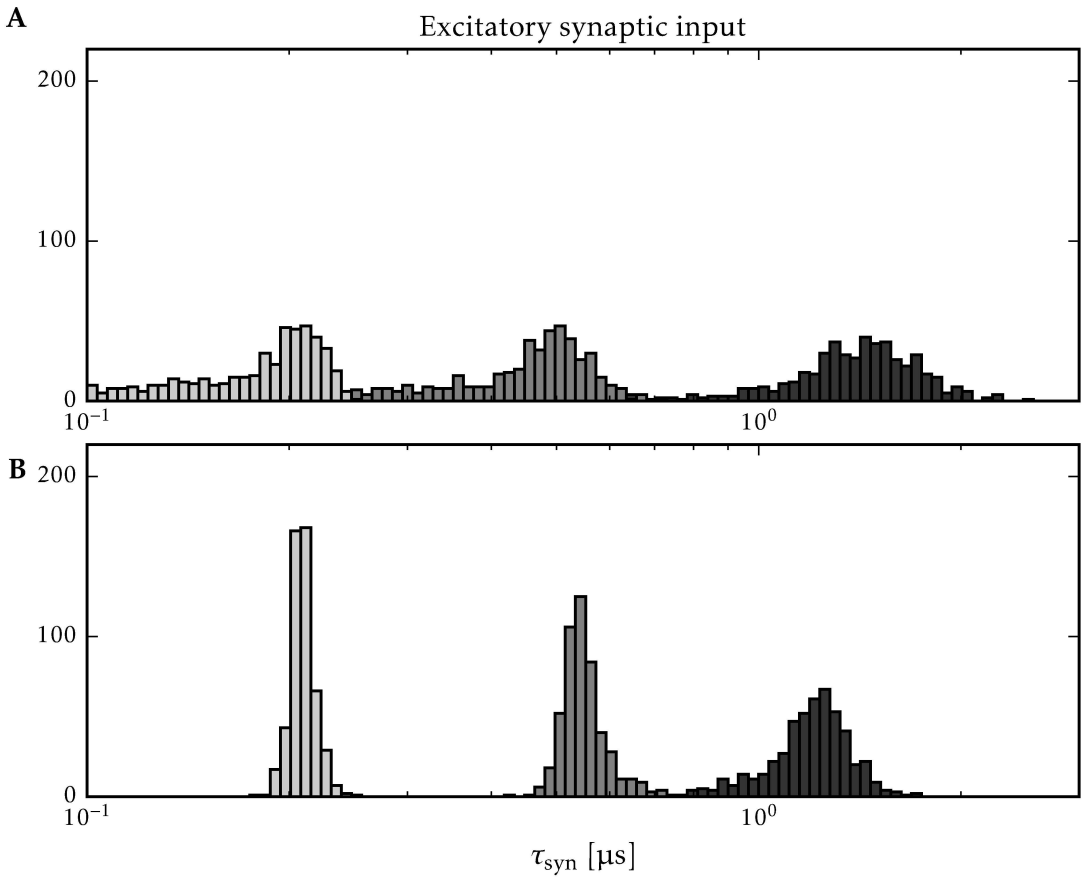
factor of 4 to 6. The final results are shown in Figure 7.25.

7.3.5 Agenda to Speed up the Calibration

So far we focused on the development and evaluation of the presented calibration methods. Therefore, the speed of the conducted measurements is not yet perfectly optimized. We currently need about 28 min to measure 11 parameter steps for 512 neurons. In the following we show approaches to improve this time.

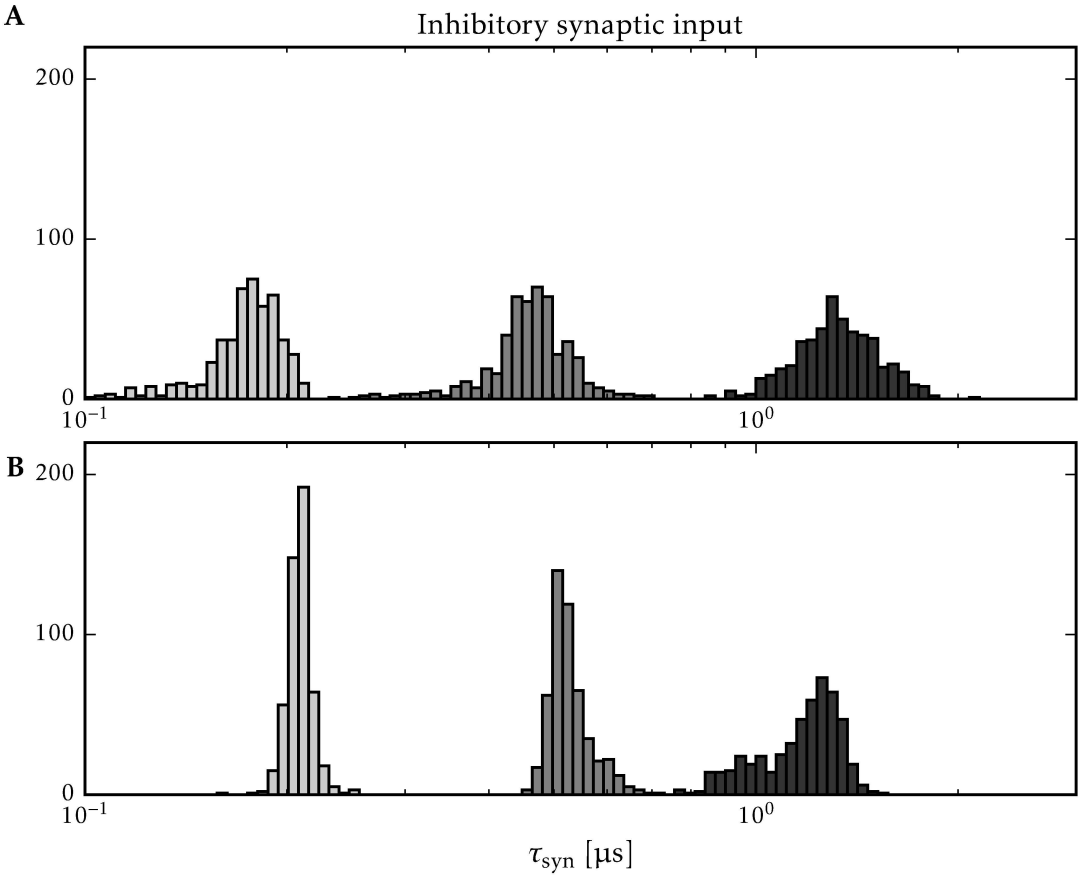
Firstly, we choose the number of steps and the number of averaged PSPs in a conservative way. For example, we use an inter-spike interval of 60 µs, but even for the largest PSPs the membrane is settled after 25 µs in the steady state, and much faster for smaller time-constants. This needs to be optimized and the inter-spike interval could be adapted to the steps. Also we average over a fairly large number of PSPs, which possibly could also be reduced. However, before we can take those measures, studies involving more HICANNs on different wafers are necessary to find a safe margin.

Secondly, the performance of the measurement process is not yet optimal. We want to estimate how long the measurement should take under ideal conditions. For this we need to take into account the data acquisition, the data transfer, the



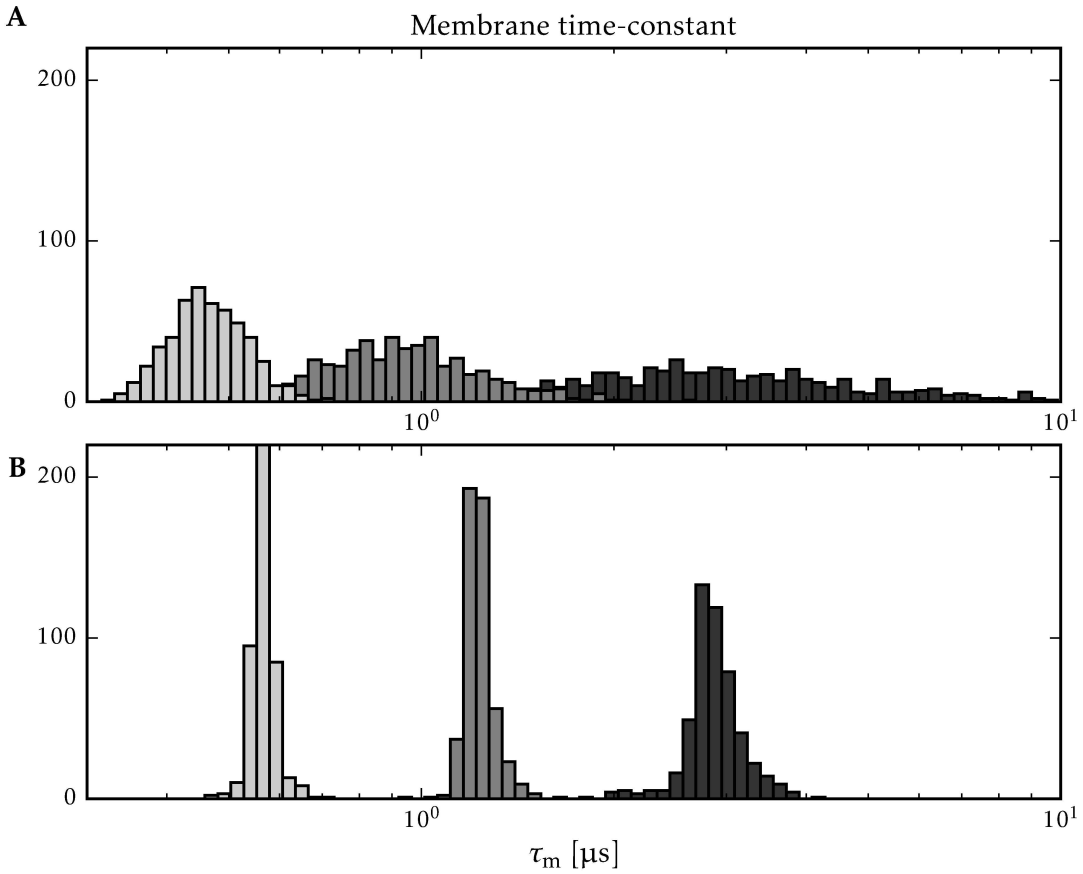
	A during calibration		**B** after calibration (N = 30)		
	V_{syntcx} [V]	$\mu \pm \sigma$ [µs]	target [µs]	$\mu \pm \sigma_m$ [µs]	$\langle \sigma_t \rangle$ [µs]
	0.41	1.39 ± 0.34	1.00	1.20 ± 0.16	0.08
	0.56	0.44 ± 0.11	0.50	0.55 ± 0.04	0.03
	0.90	0.18 ± 0.04	0.20	0.21 ± 0.01	0.01

Figure 7.23 Evaluation of the excitatory synaptic input. **A** Shows the raw data obtained during the calibration. The resulting mean time-constant μ and the standard deviation are noted in the table. **B** Shows the evaluation measurement using N = 30 repetitions. For each neuron its mean value is plotted. We compare the width σ_m of the resulting distribution, with the mean trial-to-trial variability $\langle \sigma_t \rangle$ over all neurons. For both lower values, σ_m is dominated by the trial-to-trial variations. For $\tau_{syne} = 2\,\mu s$, σ_m is twice as large as $\langle \sigma_t \rangle$. This are caused by trial-to-trial variations occurring during the calibration, that are not perfectly averaged out.



	V_{syntci} [V]	$\mu \pm \sigma$ [µs]	target [µs]	$\mu \pm \sigma_{\mathrm{m}}$ [µs]	$\langle \sigma_{\mathrm{t}} \rangle$ [µs]
	0.41	1.33 ± 0.19	1.00	1.17 ± 0.16	0.07
	0.56	0.46 ± 0.07	0.50	0.53 ± 0.04	0.03
	0.90	0.17 ± 0.02	0.20	0.21 ± 0.01	0.01

Figure 7.24 Evaluation of the inhibitory synaptic input. It confirms that the behaviour of both inputs is indeed identical. **A** Shows the raw data obtained during the calibration. The resulting mean time-constant μ and the standard deviation are noted in the table. **B** Shows the evaluation measurement using N = 30 repetitions. For each neuron its mean value is plotted. We compare the width σ_m of the resulting distribution, with the mean trial-to-trial variability $\langle \sigma_{\mathrm{t}} \rangle$ over all neurons. For both lower values σ_m is dominated by the trial-to-trial variations. For $\tau_{\mathrm{syni}} = 2\,\mu\mathrm{s}$ σ_m is twice as large as $\langle \sigma_{\mathrm{t}} \rangle$. This is caused by trial-to-trial variations occurring during the calibration, that are not perfectly averaged out.



A during calibration **B** after calibration (N = 30)

	V_m [V]	$\mu \pm \sigma$ [µs]	target [µs]	$\mu \pm \sigma_m$ [µs]	$\langle \sigma_t \rangle$ [µs]
	0.07	3.30 ± 1.88	2.00	2.87 ± 0.30	0.21
	0.36	0.98 ± 0.28	1.00	1.23 ± 0.06	0.04
	1.44	0.47 ± 0.07	0.50	0.57 ± 0.02	0.01

Figure 7.25 Evaluation of the membrane time-constant calibration. **A** Shows the raw data obtained during the calibration. The shown steps were picked, so that the mean time-constants match the target time-constants of the evaluation measurement. The resulting mean time-constant μ and the standard deviation are noted in the table. **B** Shows the evaluation measurement using N = 30 repetitions. For each neuron its mean value is plotted. We compare σ_m of the resulting distribution, with the mean trial-to-trial variability $\langle \sigma_t \rangle$ over all neurons. We notice a larger systematic error than it occurs for the synaptic time-constants. In contrast, the final σ_m reaches almost limit given by the trial-to-trial variations. For all shown values the improvement is very good.

configuration and the data processing. We begin with the data acquisition: We stimulate the neurons with 16.7 kHz and average over 200 PSPs. For the whole measurement we get

recording of a single neuron	12 ms
	× 11 steps
	× 512 neurons
gives in total a recording time of	67.584 s.

Using the transfer-rate of 300 MiB s^{-1}, which we obtained in Figure 3.4, the data transfer should take

per measurement	1 152 000 samples
	× 11 step
	× 512 neuron
gives in total	6 488 064 000 samples (≈ 12 GiB)
a data transfer time of	346 s.

The configuration currently takes

per step about	15 s
	× 11 step
gives in total	165 s.

The data processing consists mostly in averaging the PSP and fitting the PSP. This is done in parallel while the next neuron is already measured. Because we can use multiple cores, the fits can be done completely parallel to the data transfer. We therefore do not need take the fit of the PSP into account for the estimate of the total time. On the other hand, the final fit of the transformation functions is still a relevant factor and takes currently 4 min. These fits are not yet done in parallel, because this was not an issue for previous functions.

Putting all of the above estimates together, we expect a total duration for

recording	67.584 s
transfer	346 s
configuration	165 s
fit of transformation functions	240 s
gives in total	818 s or about 14 min.

This is an estimate for the fastest possible calibration speed using the same parameters. Regarding this the actual time needed – 28 min or twice of the estimate – is not particularly bad, especially as the calculated estimate here is quite optimistic. It well agrees with previous studies, that there are no obvious performance bottlenecks in our software [Mauch 2016].

However, the potential for various minor improvements exists. Parallelizing the fit of the transfer functions will gain about 3.5 min, because the currently used

machines have 8 cores, that can work in parallel. Optimizing the number of steps and the parameters of the steps can gain about 6 to 7 min, because it is likely that we can reduce the recording time for all steps to at least the half and that we can reduce the number of averaged PSPs. That leaves us with a realistic estimate of the final optimal duration of about 4 min. This would result in a real measurement time of 8 min, if no other optimization is done.

Further potential lies in the redesign of AnaRM currently done by Joscha Ilmberger. Here, an anti-aliasing low-pass filter is planned and the maximal sample rate is 33 MHz. Both will reduce the need for averaging and the transferred samples greatly. In addition, the data transfer will be using Ethernet, which will increase the transfer speed. We also could imagine, that the averaging is directly done in the FPGA of AnaRM. If this is possible, it would make the data transfer negligible, but would require a large development effort. Lastly, the playback-memory buffered configuration of the neurons will also reduce the calibration time further by reducing the configuration time.

7.3.6 Summary

We have shown that a calibration of the synaptic input can be done and leads to satisfying results. While we obtain some systematic deviation from the desired results, the limit for the precision we can reach are indeed near the trial-to-trial variations. Hereby, the later is more important, because the systematic variations can be handled for example by learning algorithms or plasticity mechanisms.

We show in Figure 7.26 a comparison to a neuron simulation done with NEST. It demonstrates well that the final variations inherent to analog neurons can be handled in neural networks simulations and that HICANN can now be used for experiment using the LIF model.

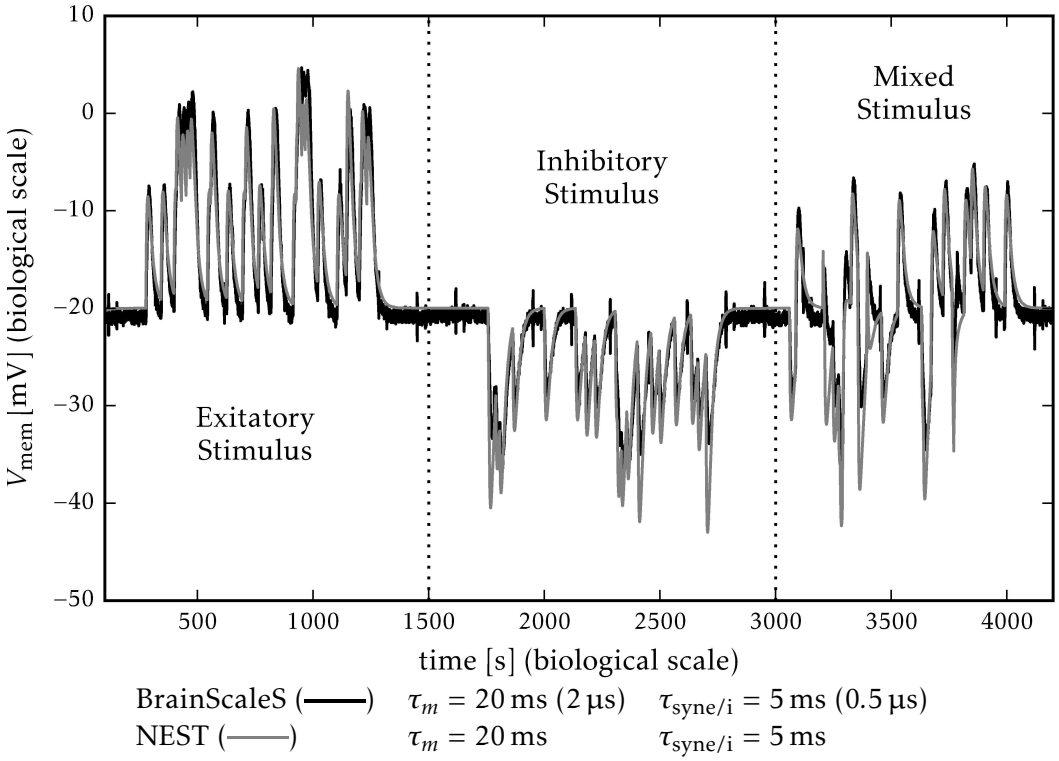


Figure 7.26 Comparision between a calibrated BrainScaleS System and the NEST simulator [Gewaltig et al. 2007]. The neurons are stimulated with possion distributed random spikes with a target rate of 2 kHz. The neuron parameters were transformed using the calibration. The synaptic weights were manually adjusted. All parameters are listed in **??**. The BrainScaleS System can be identified by the slightly noisy trace, but overall the traces match very well. Experiment and data by Johann Klähn and Sebastian Schmitt.

8 Conclusion and Outlook

In this thesis we characterized the variability in the synapses and analog neuron circuits of HICANN. We have presented a calibration of the analog neuron model of HICANN revision 4, which compensates these variations, so that the precision by which the parameters can be set is vastly improved, if an LIF neuron model is emulated. Further we characterized the strength of synapses using transistor-level Monte Carlo simulations.

We verified the calibration of the neuron potentials which were developed for revision 2. The achievable precision was limited by the current trial-to-trial variability of the analog parameters.

For the usage of the neuron circuits as LIF model emulation, the synaptic and membrane time-constants are characteristic parameters of the interaction between neurons. For these, we present a new method for calibration based on voltage recordings of PSP curves, which are caused by incoming spike events. We can extract the time-constants from a recorded PSP by fitting a parameterized model of its shape. Because recording of the neuron membrane in HICANN is afflicted by noise, we average multiple PSPs to improve the precision and verify the quality of fits using various tests. We presented the result of those measurements for a single HICANN using all of its 512 neurons. Here, we find that the achieved range of time-constants varies greatly between neurons. We find that the excitatory time-constant of a single neuron can be between $0.14 \pm 0.13\,\mu s$ and $2.3 \pm 1.1\,\mu s$, where the error indicates the standard deviation of the neuron-to-neuron variability. The common range, that can be covered by 99.7 % (corresponding to a 3σ interval of a normal distribution) of the neurons can cover, extends from 0.18 to 1.1 µs. Respectively, we find for the inhibitory time-constant that the time-constant for a single neuron lies between $0.11 \pm 0.12\,\mu s$ and $4.3 \pm 1.7\,\mu s$ and that the common range extends from 0.14 to 2.0 µs. The differences between excitatory and inhibitory time-constants are mainly caused by the asymmetry of the inhibitory and excitatory input on hardware. The precision of the calibration achieves the limit given by the trial-to-trial variability of the analog parameters. However, for time-constants above 0.5 µs the obtained precision decreases and introduces a systematic error. When fitting the model we assume that the the single data points have normally distributed variation from trial to trial. This no longer holds when the relation to the analog parameter becomes exponential, as it occurs in this range, but the distributions become skewed.

For the membrane time-constant, we employ the same method. However, here we considered only the case of a scaling of $1 : 3$ of the control parameter of the leakage conductance, so we analyzed only a subset of the available parameter space. We find the possible range for the individual neuron reaching from $0.47 \pm 0.09\,\mu s$

to $3.3 \pm 1.9\,\mu s$ and the common range extends from 0.65 to $1.14\,\mu s$. Here we can also reach a precision close to the trial-to-trial variability for the whole parameter range, but observe the same systematic error as for the synaptic time-constants, for time-constants larger than $1\,\mu s$. However, the spread of the calibrated distributions compared to the uncalibrated ones was reduced by a factor of 4 to 6.

In addition we provide a model for the strength of the synapse circuits on HICANN dependent on its digital weight w and divisor g_{div} as well as the analog reference current V_{gmax}. Since those are not accessible to direct measurements, it is based on transistor-level simulations. We find that the synapses have a weight-dependent offset, which we incorporated into the model. Monte Carlo simulations were used to estimate the expected variation in strength. We find that the variations mostly depend on the chosen divisor g_{div} and that for $g_{\mathrm{div}} < 10$ a relative error below 10 % can be expected.

In conclusion, we expect from these calibrations to improve the ability to run neural network experiments on the BrainScaleS System by increasing precision of the LIF neuron model in the hardware.

Currently the major limiting factor for the achievable precision remains the analog parameter storage. We worked with programming parameters, that were not optimized for the used systems and showed that the obtained precision is worse than the best results known by at least a factor of two. Reducing the trial-to-trial variability of the parameters further will directly improve the calibration results as well because the trial-to-trial errors of the parameters occurring during calibration affect the obtained transformations functions. We further presented the charging curves of the parameters, that identify the inhomogeneous response of the parameter during the programming process as a potential limit for the maximal achievable precision. This should be addressed by optimizing the programming of the analog parameters for each system. In case a further revision of HICANN is made, a redesign of the controller of the analog parameter storage could be considered as well, so that it can better cope with the different responses of the cells. Here the charging curves could be extended for modeling the cells responses to simulate the controller in advance.

The commission of the system and the methods for calibration we presented are restricted by the fact that only the neuron membranes can be read out from the neuron and that digital debugging features of HICANN are limited. This required us to involve methods which are costly in development time as well as in measurement time: The possibility to directly read out the integrator of the synaptic input would avoid the cumbersome deviation of the time-constants via the PSPs. Further the complexity of the integration tests reflects the difficulty to obtain debugging information from HICANN. For example, a simple spike counter would have simplified the playback memory tests. However, for these kind of features the required development time and on-chip resources have to be weighted against a possible speedup in commissioning and an increased usability of the system.

Given that the development of the BrainScaleS System is ongoing, many of our mentioned recommendations are already realized in the first prototypes of HI-

CANN's successor HICANN - Digital Learning System (HICANN-DLS). It is a entirely new implementation using a new process technology. As reaction to the difficulties in testing and commissioning of the BrainScaleS System, HICANN-DLS provides more testing features [Hartel 2016]. Thus the synaptic integrators can now be read out directly [Aamir et al. 2016]. Further, the offset and the non-linearity to the weights we describe in our model have been resolved in the redesigned synaptic circuits [Friedmann et al. 2016]. In addition, the analog parameters in HICANN-DLS have a insignificant trial-to-trial variablity [Hock 2014]. Lastly, it contains an on chip membrane ADC, which promises less noise in the readout of voltages. It is a tempting idea to explore the potential of the Plasticity Processing Unit (PPU), which is originally designed for learning tasks, for simple calibration tasks done on the chip directly. However the development of HICANN-DLS is still in the prototype phase and it will require more time and further development until it can replace the current HICANN.

Therefore the calibration tasks on HICANN should be subject to further improvements. While we could reduce trial-to-trial variability, a systematic error for larger time-constants remains. Further, our model of the synaptic strength should be extended to reflect the influence of the synapses on the conductance of the neurons. For this it only needs to be extended to incorporate the conversion stage in the synaptic input. In addition, a remaining important task is to scale the calibration routines towards a full wafer calibration. This requires two main parts: Firstly, the robustness of the methods needs to be evaluated using multiple systems to ensure that these are applicable on all HICANNs with equal results. And secondly, the performance of the calibration tools needs to be improved as well: Using buffered configuration will increase the configuration speed during calibration. Also the throughput of the measurements can be increased by recording multiple neurons in a single experiment run and switching the neuron while recording.

Lastly, we identified the transfer of the recorded data as dominant factor of the time-constant calibration. Optimizing the used parameters and steps for calibration can reduce the amount of data by at least a factor two. We also expect an improvement by the currently ongoing redesign of the AnaRMs. It will increase the number of HICANNs that can be calibrated in parallel and improve the bandwidth for the data transfer. The amount of data, which needs to be transferred, could be reduced by transferring the averaging task directly into the FPGA. This can be used not only for the PSP based calibration but also for the calibration of the membrane potentials. An ideal scenario minimizing the data transfer would be to switch through all neurons using a single playback memory configuration and then receive only the average values directly from the AnaRM.

We are looking forward to use the presented neuron calibration for neural network experiments. Those will also benefit from the suggested improvements. A deep neural network that classifies handwritten numbers of the MNIST dataset [LeCun et al. 1998], is currently developed for HICANN [Schmitt et al. 2017] and a gives first example of the calibrated neuron's potential.

A Parameter Settings

This appendix list the used hardware settings for the measurements presented in this thesis. The default setting for the neuron and the generation of spike input are described in Section 4.2.3. For all measurements the analog output channel 0 of the HICANNs were used. The list here does not include all experiments, for simpler ones the setting are noted in the main part of the thesis.

A.1 Operating Voltages

All voltage were left at their default settings after powering the system. We changed only VOH an VOL for the system. Especially for HICANN revision 2 this is essential to operate the synapse drivers correctly.

BrainScaleS System: W_G04 (HICANN revision 2):

DI_VBIAS_LVDS = 1.74 V, DI_VCC = 1.76 V, DI_VCC33ANA = 3.25 V, DI_VCCANA_PLL = 1.75 V, VDD12 = 10.71 V, VDD25 = 2.53 V, VDD5 = 5.22 V, VDDA_VDDPLL = 1.89 V, VDDBUS = 1.00 V, VDD_VDDOUT = 1.90 V, VOH = 1.00 V, VOL = 0.61 V

BrainScaleS System: W_G06 (HICANN revision 4.1):

DI_VBIAS_LVDS = 1.24 V, DI_VCC = 1.87 V, DI_VCC33ANA = 3.26 V, DI_VCCANA_PLL = 1.85 V, VDD12 = 10.94 V, VDD25 = 2.54 V, VDD5 = 5.20 V, VDDA_VDDPLL = 1.97 V, VDDBUS = 1.00 V, VDD_VDDOUT = 1.99 V, VOH = 1.65 V, VOL = 1.42 V

We could not obtain these currently due to a machine failure. These will be included in the finally printed version.

A.2 Floating Gate Controller

Programming scheme of the floating gate controller:

	revision 2				revision 4 and 4.1				
direction	down	up	down	up	down[a]	up[bc]	down[c]	up[c]	up[d]
maxcycle	255		10		127	255	10		127
readtime	40		63		63	40	63		63
pulselength	9		1		63	9	1		63
voltagewritetime	15		5		63	15	5		63
currentwritetime	1		20		63	1	20		63
accelartorstep	9		63		2	9	63		2
fg_bias	8				0				
fg_biasn	5				0				

[a] Write all floating gate values to zero

[b] Before writing any other cells this step is done once only to write the global int_op_bias parameter to its value

[c] Normal programming

[d]Fast upwards programming applied to current parameter, where all values in the line are above 800.

We want to especially thank Paul Müller for intensively testing and improving the configuration for revison 4 and 4.1 by adding the special handling for the int_op_bias.

A.3 Operational software

The used software is state is archived with the raw data of our measurements. We document: For all used repositories the currently checked out commit, untracked files and changes on tracked files. The set environment variables. The loaded modules. The use waf version. The hostname. The used python packages (via the anconda packet manger). Installed system package.

A.4 HICANN Revision 2 Measurements

A.4.1 Default Values

Neuron parameters:

$I_{\text{breset}} = 1023\ DAC$	$I_{\text{bstim}} = 1023\ DAC$	$V_{\text{bexp}} = 2.500\,\mu\text{A}$	$V_{\text{bout}} = 0.750\,\mu\text{A}$
$V_{\text{br}} = 0\ DAC$	$V_{\text{bstdf}} = 0\ DAC$	$V_{\text{ccas}} = 800\ DAC$	$V_{\text{clra}} = 0\ DAC$
$V_{\text{clrc}} = 0\ DAC$	$V_{\text{dep}} = 0\ DAC$	$V_{\text{dllres}} = 200\ DAC$	$V_{\text{dtc}} = 0\ DAC$
$V_{\text{fac}} = 0\ DAC$	$V_{\text{gmax0}} = 0.080\,\text{V}$	$V_{\text{gmax1}} = 0.080\,\text{V}$	$V_{\text{gmax2}} = 0.080\,\text{V}$
$V_{\text{gmax3}} = 0.080\,\text{V}$	$V_{\text{m}} = 0\ DAC$	$V_{\text{reset}} = 0.500\,\text{V}$	$V_{\text{stdf}} = 0\ DAC$
$V_{\text{thigh}} = 0\ DAC$	$V_{\text{tlow}} = 0\ DAC$	$\text{int\_op\_bias} = 1023\ DAC$	

Shared parameters:

$E_{\text{l}} = 0.800\,\text{V}$	$E_{\text{syni}} = 0.600\,\text{V}$	$E_{\text{synx}} = 1.300\,\text{V}$	$I_{\text{bexp}} = 2.500\,\mu\text{A}$
$I_{\text{convi}} = 0.625\,\mu\text{A}$	$I_{\text{convx}} = 0.625\,\mu\text{A}$	$I_{\text{fire}} = 0.000\,\mu\text{A}$	$I_{\text{gl}} = 1.000\,\mu\text{A}$
$I_{\text{gladapt}} = 0.000\,\mu\text{A}$	$I_{\text{intbbi}} = 2.000\,\mu\text{A}$	$I_{\text{intbbx}} = 2.000\,\mu\text{A}$	$I_{\text{pl}} = 2.000\,\mu\text{A}$
$I_{\text{radapt}} = 2.500\,\mu\text{A}$	$I_{\text{rexp}} = 2.500\,\mu\text{A}$	$I_{\text{spikeamp}} = 2.000\,\mu\text{A}$	$V_{\text{convoffi}} = 1.800\,\text{V}$
$V_{\text{convoffx}} = 1.800\,\text{V}$	$V_{\text{exp}} = 1.800\,\text{V}$	$V_{\text{syni}} = 0.900\,\text{V}$	$V_{\text{syntci}} = 1.420\,\text{V}$
$V_{\text{syntcx}} = 1.420\,\text{V}$	$V_{\text{synx}} = 0.900\,\text{V}$	$V_{\text{t}} = 1.000\,\text{V}$	

A.4.2 Synaptic Time-Constant Characterization

Step parameters: $V_{\text{syntcx}} = 1.150, 1.175, 1.200, 1.225, 1.250, 1.275, 1.300, 1.325, 1.350, 1.375, 1.400, 1.425, 1.450, 1.475, 1.500, 1.525, 1.550, 1.575$ and $1.600\,\text{V}$

Non-default parameters: $E_{\text{l}} = 0.8\,\text{V}$ (calibrated), $E_{\text{syni}} = 0.65\,\text{V}$ (calibrated), $E_{\text{synx}} = 0.95\,\text{V}$ (calibrated), $V_{\text{t}} = 1.2\,\text{V}$ (calibrated), $V_{\text{gmax}} = 0.14\,\mu\text{A}$ (calibrated), $V_{\text{reset}} = 0.3\,\text{V}$ (calibrated), $I_{\text{gl}} = 0.5\,\mu\text{A}$ (calibrated), $f_{\text{stim}} = 4.726\,56\,\text{kHz}$, $g_{\text{div}} = 4$ and $w = 15$.

A.5 HICANN Revision 4.1 Measurements

Default Values

Neuron parameters:

$I_{\text{breset}} = 1023\ DAC$	$I_{\text{bstim}} = 1023\ DAC$	$V_{\text{bexp}} = 2.500\,\mu\text{A}$	$V_{\text{bout}} = 0.750\,\mu\text{A}$
$V_{\text{br}} = 0\ DAC$	$V_{\text{bstdf}} = 0\ DAC$	$V_{\text{ccas}} = 800\ DAC$	$V_{\text{clra}} = 0\ DAC$
$V_{\text{clrc}} = 0\ DAC$	$V_{\text{dep}} = 0\ DAC$	$V_{\text{dllres}} = 200\ DAC$	$V_{\text{dtc}} = 0\ DAC$
$V_{\text{fac}} = 0\ DAC$	$V_{\text{gmax0}} = 0.080\,\text{V}$	$V_{\text{gmax1}} = 0.080\,\text{V}$	$V_{\text{gmax2}} = 0.080\,\text{V}$
$V_{\text{gmax3}} = 0.080\,\text{V}$	$V_{\text{m}} = 0\ DAC$	$V_{\text{reset}} = 0.500\,\text{V}$	$V_{\text{stdf}} = 0\ DAC$
$V_{\text{thigh}} = 0\ DAC$	$V_{\text{tlow}} = 0\ DAC$	int_op_bias $= 1023\ DAC$	

In all cases a scaling of 1 : 3 is used for I_{gl}.
Shared parameters:

$E_{\text{l}} = 0.800\,\text{V}$	$E_{\text{syni}} = 0.600\,\text{V}$	$E_{\text{synx}} = 1.300\,\text{V}$	$I_{\text{bexp}} = 2.500\,\mu\text{A}$
$I_{\text{convi}} = 0.625\,\mu\text{A}$	$I_{\text{convx}} = 0.625\,\mu\text{A}$	$I_{\text{fire}} = 0.000\,\mu\text{A}$	$I_{\text{gl}} = 1.000\,\mu\text{A}$
$I_{\text{gladapt}} = 0.000\,\mu\text{A}$	$I_{\text{intbbi}} = 2.000\,\mu\text{A}$	$I_{\text{intbbx}} = 2.000\,\mu\text{A}$	$I_{\text{pl}} = 2.000\,\mu\text{A}$
$I_{\text{radapt}} = 2.500\,\mu\text{A}$	$I_{\text{rexp}} = 2.500\,\mu\text{A}$	$I_{\text{spikeamp}} = 2.000\,\mu\text{A}$	$V_{\text{convoffi}} = 1.800\,\text{V}$
$V_{\text{convoffx}} = 1.800\,\text{V}$	$V_{\text{exp}} = 1.800\,\text{V}$	$V_{\text{syni}} = 0.900\,\text{V}$	$V_{\text{syntci}} = 1.420\,\text{V}$
$V_{\text{syntcx}} = 1.420\,\text{V}$	$V_{\text{synx}} = 0.900\,\text{V}$	$V_{\text{t}} = 1.000\,\text{V}$	

Resting Potential Calibration

Step parameters: $E_{\text{l}} = 0.7$, 0.8 and 0.9 V
Non-default parameters: $V_{\text{reset}} = 0.900\,\text{V}$ (calibrated), $V_{\text{t}} = 1.200\,\text{V}$ (calibrated), $I_{\text{convi}} = 0.000\,\mu\text{A}$, $I_{\text{convx}} = 0.000\,\mu\text{A}$

Reset Potential Calibration

Step parameters:

		1	2	3	4
E_{l}	[V]	0.9	1.0	1.1	1.2
V_{reset}	[V]	0.5	0.6	0.7	0.8
V_{t}	[V]	0.7	0.8	0.9	1.0

Non-default parameters: $I_{\text{pl}} = 0.020\,\mu\text{A}$, $I_{\text{convi}} = 0.000\,\mu\text{A}$, $I_{\text{convx}} = 0.000\,\mu\text{A}$, $I_{\text{gl}} = 1.100\,\mu\text{A}$

A.5.1 Threshold Potential Calibration

Step parameters:

		1	2	3	4	5
E_l	[V]	0.9	1.0	1.1	1.2	1.3
V_{reset}	[V]	0.5	0.6	0.7	0.8	0.9
V_t	[V]	0.7	0.8	0.9	1.0	1.1

Non-default parameters: $I_{convi} = 0.000\,\mu A$, $I_{convx} = 0.000\,\mu A$, $I_{gl} = 1.500\,\mu A$

A.5.2 Inhibitory Potential Calibration

Step parameters: $E_l = 0.4$, 0.6 and 0.8 V

Non-default parameters: $E_l = 0.800$ V (calibrated), $V_{syntci} = 1.800$ V, $V_{syntcx} = 1.800$ V, $V_t = 1.200$ V (calibrated), $I_{convx} = 0.000\,\mu A$, $V_{reset} = 0.900$ V (calibrated), $I_{gl} = 0.000\,\mu A$, $V_{convoffi} = 0.100$ V

A.5.3 Bias Generator Calibration

Step parameters (only for the calibrated input):

$V_{convoffi}$ or $V_{convoffx} = 0$ to 1.8 V in steps of 0.03 V

Non-default parameters: $E_l = 0.8$ V (calibrated), $E_{syni} = 0.2$ V (calibrated), $E_{synx} = 1.4$ V, $V_t = 1.2$ V (calibrated), $V_{reset} = 0.4$ V (calibrated), $I_{convi} = 0$ uA (only for $V_{convoffx}$), $I_{convx} = 0$ uA (only for $V_{convoffi}$)

A.5.4 Synaptic Time-Constant Calibration

Step parameters for inhibitory input:

$V_{syntci} = 0.30$, 0.35, 0.41, 0.48, 0.56, 0.66, 0.77, 0.90, 1.20, 1.50 and 1.80 V

Step parameters for excitatory input:

$V_{syntcx} = 0.30$, 0.35, 0.41, 0.48, 0.56, 0.66, 0.77, 0.90, 1.20, 1.50 and 1.80 V

Non-default parameters: $E_l = 0.800$ V (calibrated), $E_{syni} = 0.600$ V (calibrated), $E_{synx} = 1.200$ V, $V_t = 1.200$ V (calibrated), $V_{convoffi} = 1.800$ V (calibrated), $V_{convoffx} = 1.800$ V (calibrated), $I_{gl} = 0.400\,\mu A$, $g_{div} = 30$, $V_{gmax0} = 0.050$ V, $V_{reset} = 0.300$ V (calibrated), $f_{stim} = 16.6$ kHz

A.5.5 Membrane Time-Constant Calibration

Step parameters: $I_{gl} = 0.10$, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 1.20 and 2.00 µA

Non-default parameters: $E_l = 0.800$ V (calibrated), $E_{syni} = 0.600$ V (calibrated), $E_{synx} = 1.300$ V, $V_{reset} = 0.300$ V (calibrated), $V_{gmax0} = 1.000$ V, $V_{syntcx} = 1.600$ V, $V_t = 1.200$ V (calibrated), $V_{convoffi} = 0.900$ V (calibrated), $V_{convoffx} = 0.900$ V (calibrated), $g_{div} = 16$, $w = 15$

A.5.6 Evaluation of V_{syntcx}, V_{syntc} and I_{gl}

V_{syntcx} step parameters: $V_{syntcx} = 1.0$, 0.5 and 0.2 µs (calibrated)

Non-default parameters: $E_l = 0.800$ V (calibrated), $E_{syni} = 0.600$ V (calibrated), $E_{synx} = 1.200$ V, $V_t = 1.200$ V (calibrated), $V_{convoffi} = 1.800$ V (calibrated), $V_{convoffx} = 1.800$ V (calibrated), $I_{gl} = 0.400\,\mu A$, $V_{gmax0} = 0.050$ V, $V_{reset} = 0.300$ V (calibrated), $g_{div} = 30$, $w = 15$, -$f_{stim} = 16.6$ kHz

V_{syntci} step parameters: $V_{syntci} = 1.0$, 0.5 and 0.2 µs (calibrated)

Non-default parameters: $E_{\mathrm{l}} = 0.800\,\mathrm{V}$ (calibrated), $E_{\mathrm{syni}} = 0.600\,\mathrm{V}$ (calibrated), $E_{\mathrm{synx}} = 1.200\,\mathrm{V}$, $V_{\mathrm{t}} = 1.200\,\mathrm{V}$ (calibrated), $V_{\mathrm{convoffi}} = 1.800\,\mathrm{V}$ (calibrated), $V_{\mathrm{convoffx}} = 1.800\,\mathrm{V}$ (calibrated), $I_{\mathrm{gl}} = 0.400\,\mu\mathrm{A}$, $V_{\mathrm{gmax0}} = 0.050\,\mathrm{V}$, $V_{\mathrm{reset}} = 0.300\,\mathrm{V}$ (calibrated), $g_{\mathrm{div}} = 30$, $w = 15$, $f_{\mathrm{stim}} = 16.6\,\mathrm{kHz}$

I_{gl} step parameters: $I_{\mathrm{gl}} = 3.0$, 2.0, 1.0 and $0.5\,\mu\mathrm{A}$ (calibrated)

Non-default parameters: $E_{\mathrm{l}} = 0.800\,\mathrm{V}$ (calibrated), $E_{\mathrm{syni}} = 0.600\,\mathrm{V}$ (calibrated), $E_{\mathrm{synx}} = 1.300\,\mathrm{V}$, $V_{\mathrm{reset}} = 0.300\,\mathrm{V}$ (calibrated), $V_{\mathrm{gmax0}} = 1.000\,\mathrm{V}$, $V_{\mathrm{syntcx}} = 1.600\,\mathrm{V}$, $V_{\mathrm{t}} = 1.200\,\mathrm{V}$ (calibrated), $V_{\mathrm{convoffi}} = 0.900\,\mathrm{V}$ (calibrated), $V_{\mathrm{convoffx}} = 0.900\,\mathrm{V}$ (calibrated), $g_{\mathrm{div}} = 16$, $w = 15$, $f_{\mathrm{stim}} = 16.6\,\mathrm{kHz}$

A.5.7 Synaptic Weight and Divisor Example

For this measurements 3 synapses driver were directly connected (mirrored) to use 3 synapse per neuron in parallel.

Common non-default parameters:

$E_{\mathrm{l}} = 0.800\,\mathrm{V}$ (calibrated), $E_{\mathrm{syni}} = 0.600\,\mathrm{V}$ (calibrated), $E_{\mathrm{synx}} = 1.200\,\mathrm{V}$, $I_{\mathrm{gl}} = 2.0\,\mu\mathrm{s}$ (calibrated), $V_{\mathrm{reset}} = 0.300\,\mathrm{V}$ (calibrated), $V_{\mathrm{syntcx}} = 0.5\,\mu\mathrm{s}$ (calibrated), $V_{\mathrm{convoffi}} = 0.001\,\mathrm{mV}$ (calibrated), $V_{\mathrm{convoffx}} = 0.001\,\mathrm{mV}$ (calibrated), $V_{\mathrm{t}} = 1.200\,\mathrm{V}$ (calibrated), $f_{\mathrm{stim}} = 24.324\,98\,\mathrm{kHz}$

Weight w measurement step parameters: $w = 0.15$ to in steps of 1

Weight measurement non-default parameters: $V_{\mathrm{gmax0}} = 0.2\,\mathrm{V}$ and $g_{\mathrm{div}} = 30$

Divisor measurement step parameters: $w = 2.30$ to in steps of 2

Divisor measurement non-default parameters: $V_{\mathrm{gmax0}} = 0.7\,\mathrm{V}$ and $w = 3$

A.5.8 Default Parameters of the Simulations

Monte Carlo simulations were conducted by setting *sigma* = 3 in the model.

Simplified Neuron Simulation:

$E_{\mathrm{l}} = 0.8\,\mathrm{V}$, $E_{\mathrm{syni}} = 0.6\,\mathrm{V}$, $E_{\mathrm{synx}} = 1.0\,\mathrm{V}$, $I_{\mathrm{convi}} = 2.5\,\mathrm{nA}$, $I_{\mathrm{convx}} = 2.5\,\mathrm{nA}$, $I_{\mathrm{gl}} = 0.1\,\mathrm{nA}$, $I_{\mathrm{intbbi}} = 2.0\,\mathrm{nA}$, and $I_{\mathrm{intbbx}} = 2.0\,\mathrm{nA}$.

Simplified Neuron Simulation with Synapses:

$E_{\mathrm{l}} = 0.9\,\mathrm{V}$, $E_{\mathrm{syni}} = 0.8\,\mathrm{V}$, $E_{\mathrm{synx}} = 1.0\,\mathrm{V}$, $I_{\mathrm{convi}} = 2.0\,\mathrm{nA}$, $I_{\mathrm{convx}} = 2.0\,\mathrm{nA}$, $I_{\mathrm{gl}} = 2.0\,\mathrm{nA}$, $I_{\mathrm{intbbi}} = 2.0\,\mathrm{nA}$, $I_{\mathrm{intbbx}} = 2.0\,\mathrm{nA}$, $V_{\mathrm{convoffi}} = 1.0\,\mathrm{V}$, $V_{\mathrm{convoffx}} = 1.0\,\mathrm{V}$, $V_{\mathrm{gmax}} = 2\,\mu\mathrm{A}$, $V_{\mathrm{syni}} = 1.0\,\mathrm{V}$, $V_{\mathrm{synx}} = 1.0\,\mathrm{V}$, $V_{\mathrm{syntci}} = 1.1\,\mathrm{V}$, $V_{\mathrm{syntcx}} = 1.10\,\mathrm{V}$, $w = 15$, $g_{\mathrm{div}} = 2$ and PLL = 100 MHz

Complete Neuron Simulation for the Calibration Framework:

As given in Appendices A.4.1 and A.5.

B Used Hardware

We conducted all measurements presented in this work on BrainScaleS Systems. For both the results presented for HICANN revision 2 as well as for HICANN revision 4 we selected a HICANN that showed good results in the calibration of its neuron potentials. However, we did not found any significant difference to other HICANNs with similar premises. Description of the abbreviations can found in the *Neuromorphic Platform Specification* [SP9 Spec 2015]. We here list the used hardware components:

revision	Component	Description
2	MainPCB and Wafer	W_G04
	HICANN	257
	FCP	45[1]
	AnaRM	Analog 0: B291681[2]
		Analog 1: B201280[3]
	AnaB	B31/B59[4]
	AuxPwr	
	PowerIt	
	Cure	0x19fe7b, 0x1a64c8, 0x19fe9c, 0x19b753, 0x19d087, 0x1a0ec3, 0x19ad67, 0x1a5cbc
4.1	MainPCB and Wafer	W_G06
	HICANN	95
	FCP	20[5]
	AnaRM	Analog 0: B291693[3]
		Analog 1: B201251[3]
	AnaB	B28/B58[4]
	AuxPwr	B25/B26
	Cure	0x19fe7b, 0x1a64c8, 0x19fe9c, 0x19b753, 0x19d087, 0x1a0ec3, 0x19ad67, 0x1a5cbc
	PowerIt	B19

[1] bitfile: hmf-fpga-top_master_2016-08-02_aca3684.bin serialnumber: 0x12280c7a8e51982e12
[2] bitfile: rev. 10d79fae
[3] bitfile: rev. 1bbb73a
[4] Shielded switching Regulators for 5V
[5] bitfile: hmf-fpga-top_master_2016-08-02_aca3684.bin serialnumber: 0x12280c7a8e51983416

C HICANN Parameters and Circuit Components

We here list the meaning of all currents, voltages, digital parameters and components of HICANN that we name in this work. The names try to follow the notation given in previous works.

E_{l} Resting potential of the neuron

E_{syni} Inhibitory reversal potential of the neuron

E_{synx} Excitatory reversal potential of the neuron

E_{syn} E_{syni} or E_{synx}

I_{OTA_1} Output current of OTA_1

I_{convi} Bias current for OTA_1 in the inhibitory input

I_{convx} Bias current for OTA_1 in the excitatory input

I_{conv} I_{convi} or I_{convx}

I_{gl} Bias current controlling the leakage conductance of the membrane

I_{pl} Bias current controlling the refractory period neuron

I_{syn} Input current from the synapses to the integrator of the synaptic input

R_{syntc} Resistor in the synaptic integrator, controlled by V_{syntc}

V_{convoffi} Bias voltage controlling the offset compensation for OTA_1 in the inhibitory synaptic input

V_{convoffx} Bias voltage controlling the offset compensation for OTA_1 in the excitatory synaptic input

V_{convoff} V_{convoffi} or V_{convoffx}

V_{gmax} Reference current(!) for synapses modified by the synaptic weight and the divisor in the synapse driver

$V_{\mathsf{integrator}}$ Internal voltage of the integrator in the synaptic input

V_{mem} Membrane voltage of the neuron

V_{reset} Control voltage for the reset potential of the neuron

V_{syni} Reference voltage for the inhibitory synaptic input

V_{syntci} Control voltage for the inhibitory synaptic time-constant

V_{syntcx} Control voltage for the excitatory synaptic time-constant

V_{syntc} V_{syntci} or V_{syntcx}

V_{synx} Reference voltage for the excitatory synaptic input

V_{syn} V_{syni} or V_{synx}

V_{t} Spike threshold of the neuron

τ_{syn} Time-constant of the synaptic integrator.

OTA_0 OTA of the synaptic input implementing the reversal potential [Millner 2012]

OTA_1 OTA of the synaptic input translating the integrated synapse pulse into a bias current for OTA_0

R_0 Resistor between membrane and reverale potential potential. Replaced OTA_0 in revision 4.

VDD12 floating-gate programming supply

VDD25 floating-gate supply

g_{div} Divisior applied on V_{gmax} in the synapse driver. Valid values range from 1 to 30.

g_{scale} Fixed scaling factor for the reference current V_{gmax}.

t_{syn} Duration of an synaptic event.

w Digital synaptic weight. Valid values range from 0 to 15.

D Acronyms

ADC Analog-to-Digital Converter

AdEx Adaptive Exponential Integrate-and-Fire

AER address–event representation

AnaRM Analog Readout Module

BSIM Berkeley Short-channel IGFET Model [Cheng et al. 1996]

CGL control gate large

CGS control gate small

CMOS Complementary Metal-Oxide-Semiconductor

DAC Digital-to-Analog Converter .

DLL Delay-Locked Loop

DNC Digital Network Chip

ESS Executable System Specification

FCP FPGA Communication PCB

FPGA Field-Programmable Gate Array

HALbe Hardware Abstraction Layer Backend

HBP Human Brain Project

HICANN High Input Count Analog Neuronal Network

HICANN-ARQ HICANN ARQ protocol

HICANN-DLS HICANN - Digital Learning System

HostARQ Host ARQ protocol

IP Internet Protocol

JTAG Joint Test Action Group

LIF Leaky Integrate-and-Fire

MCC Multi-Compartment Chip

NDA Non Disclosure Agreement

NM-PM1 Neuromorphic Physical Model version 1

OP operational amplifier

OTA operational transconductance amplifier

PCS pulse communication subgroup

PLL Phase-Locked Loop

PPU Plasticity Processing Unit

PSP Postsynaptic Potential

SRAM Static Random Access Memory

STDP Spike Timing Dependent Plasticity

StHAL Stateful Hardware Abstraction Layer

STP Short Term Plasticity

VLSI very-large-scale integration

E Third Party Software

Boost.Python A C++ library which enables seamless interoperability between C++ and the Python programming language version 1.49.0 by *Dave Abrahams* under *Boost Software License - Version 1.0* [http://www.boost.org/doc/libs/1_49_0/libs/python/doc/index.html][Abrahams et al. 2003]

C++ C++ Programming Language, using the C++11 standard [Stroustrup 2013]

Jupyter Notebook version 5.0.0 by *Project Jupyter* under *BSD license* [http://jupyter.org]

LMFIT Non-Linear Least-Square Minimization and Curve-Fitting for Python, version 0.9.3, by *Matthew Newville, Till Stensitzki* under *MIT license* [http://lmfit.github.io/lmfit-py/]

matplotlib version 1.5.4 by *The matplotlib development team* under *custom license* [Hunter 2007]

NumPY Version 1.11.1 by *Numpy Developers* under *BSD license* [http://www.numpy.org/]

pandas version 0.18.1 by *PyData Development Team* under *BSD license* [http://pandas.pydata.org]

Py++ Py++ - Boost.Python code generator, by *Roman Yakovenko* under *Boost Software License - Version 1.0* [https://sourceforge.net/projects/pygccxml/]

Python The python programming language, version 2.7.12 by *Python Software Foundation* under *Python Software Foundation License*[http://www.python.org]

PyUblas version 2013.1 by *Andreas Klöckner* under *BSD License* [https://mathema.tician.de/software/pyublas/]

SciPy Version 0.17.1 by *SciPy Developers* under *BSD license* [http://www.scipy.org/]

Spectre Circuit Simulator version 10.1.1.047.isr4 32bit – 19 Jan 2011 by *Cadence* under *closed license*[https://www.cadence.com/content/cadence-www/global/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-circuit-simu
html]

F Bibliography

Aamir, Syed A., Paul Müller, Andreas Hartel, Johannes Schemmel, and Karlheinz Meier (2016). "A highly tunable 65-nm CMOS LIF neuron for a large scale neuromorphic system". In: *IEEE European Solid-State Circuits Conference (ESSCIRC)*, pp. 71–74. DOI: 10.1109/ESSCIRC.2016.7598245 (cit. on p. 145).

Abrahams, David and Ralf W. Grosse-Kunstleve (2003). *Building Hybrid Systems with Boost.Python*. URL: http://www.boostpro.com/writing/bpl.pdf (cit. on p. 159).

Alberts, Bruce, Alxander Johnson, Julian H. Lewis, Martin Raff, Keith Roberts, and Peter Walter (2007). *Molecular Biology of the Cell*. Taylor & Francis Group. ISBN: 9781136844423 (cit. on p. 5).

Alevi, Denis (2015). "Investigating Competitive Dynamics in a Recurrent Neural Network on Neuromorphic Hardware". Bachelor thesis. Universität Heidelberg (cit. on p. 105).

Amir, Arnon, Piyali Datta, William P. Risk, Andrew S. Cassidy, Jeffrey Kusnitz, Steven K. Esser, Alexander Andreopoulos, Theodore M. Wong, Myron Flickner, and Rodrigo Alvarez-Icaza (2013). "Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores". In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, pp. 1–10. DOI: 10.1109/IJCNN.2013.6707078 (cit. on p. 2).

Azevedo, Frederico A.C., Ludmila R.B. Carvalho, Lea T. Grinberg, José Marcelo Farfel, Renata E.L. Ferretti, Renata E.P. Leite, Roberto Lent, Suzana Herculano-Houzel, et al. (2009). "Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain". In: *Journal of Comparative Neurology* 513.5, pp. 532–541. DOI: 10.1002/cne.21974 (cit. on p. 1).

Benjamin, Ben V., Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V. Arthur, Paul A. Merolla, and Kwabena Boahen (2014). "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations". In: *Proceedings of the IEEE* 102.5, pp. 699–716. ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2313565 (cit. on p. 3).

Billaudelle, Sebastian (2014). "Characterisation and Calibration of Short Term Plasticity on a Neuromorphic Hardware Chip". Bachelor thesis. Universität Heidelberg (cit. on pp. 16, 35).

Brette, Romain and Wulfram Gerstner (2005). "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity". In: *Journal of neurophysiology* 94.5, pp. 3637–3642. DOI: 10.1152/jn.00686.2005 (cit. on pp. 3, 5).

Brüderle, Daniel, Mihai A. Petrovici, Bernhard Vogginger, Matthias Ehrlich, Thomas Pfeil, Sebastian Millner, Andreas Grübl, Karsten Wendt, Eric Müller, Marc-Olivier Schwartz, and et al. (2011). "A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems". In: *Biological Cybernetics* 104, pp. 263–296. DOI: 10.1007/s00422-011-0435-9 (cit. on pp. 3, 23, 25, 27).

Bytschok, Ilja (2011). "From Shared Input to correlated Neuron Dynamics: Development of a Predictive Framework". Diploma thesis. Universität Heidelberg (cit. on p. 8).

Cassidy, Andrew S., Paul A. Merolla, John V. Arthur, Steven K. Esser, Bryan L. Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada, Theodore M. Wong, Vitaly Feldman, Arnon Amir, Daniel B.-D. Rubin, Filipp Akopyan, Emmett McQuinn, William P. Risk, and Dharmendra S. Modha (2013). "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores". In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–10. DOI: 10.1109/IJCNN.2013.6707077 (cit. on p. 2).

Cheng, Yuhua, Mansun Chan, Kelvin Hui, Min-chie Jeng, Zhihong Liu, Jianhui Huang, Kai Chen, James Chen, Robert Tu, Ping K Ko, et al. (1996). "BSIM3v3 manual". In: *University of California, Berkeley* (cit. on pp. 53, 157).

Davison, Andrew, Daniel Brüderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger (2009). "PyNN: a common interface for neuronal network simulators". In: *Frontiers in Neuroinformatics* 2, p. 11. ISSN: 1662-5196. DOI: 10.3389/neuro.11.011.2008 (cit. on p. 23).

Dayan, Peter and Laurence F. Abbott (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambride, Massachusetts: The MIT press. ISBN: 0-262-04199-5. DOI: 10.1080/09515080210000042076 (cit. on p. 5).

Diamond, Alan, Thomas Nowotny, and Michael Schmuker (2016). "Comparing neuromorphic solutions in action: implementing a bio-inspired solution to a benchmark classification task on three parallel-computing platforms". In: *Frontiers in Neuroscience* 9.491. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00491 (cit. on p. 1).

Dugas, Charles, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia (2001). "Incorporating second-order functional knowledge for better option pricing". In: *Advances in neural information processing systems*, pp. 472–478. URL: http://www.iro.umontreal.ca/~lisa/pointeurs/nips00-convex.ps (cit. on p. 109).

Epp, Bernard (2016). "Charakterisierung und Vergleich von zwei ADC-Modulen". Bachelor thesis. Universität Heidelberg (cit. on p. 30).

Esser, Steven K., Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha (2016). "Convolutional Networks for Fast, Energy-Efficient Neuromorphic Computing". In: *Proceedings*

of the National Academy of Sciences 113.41, pp. 11441–11446. DOI: 10.1073/pnas.1604850113 (cit. on p. 2).

Esser, Steven K., Alexander Andreopoulos, Rathinakumar Appuswamy, Piyali Datta, Davis Barch, Arnon Amir, John Arthur, Alex Cassidy, Myron Flickner, Paul Merolla, et al. (2013). "Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores". In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, pp. 1–10. DOI: 10.1109/IJCNN.2013.6706746 (cit. on p. 2).

Friedmann, Simon, Johannes Schemmel, Andreas Grübl, Andreas Hartel, Matthias Hock, and Karlheinz Meier (2016). "Demonstrating Hybrid Learning in a Flexible Neuromorphic Hardware System". In: *IEEE Transactions on Biomedical Circuits and Systems* PP.99, pp. 1–15. ISSN: 1932-4545. DOI: 10.1109/TBCAS.2016.2579164 (cit. on pp. 79, 145).

Friedrich, Arno (2015). "Charakterisierung von Adaption auf neuromorpher Hardware". Master thesis. Universität Heidelberg (cit. on p. 15).

Furber, Steve (2016). "Large-scale neuromorphic computing systems". In: *Journal of Neural Engineering* 13.5, p. 051001. DOI: 10.1088/1741-2560/13/5/051001 (cit. on pp. 2–3, 23).

Furber, Steve B., Francesco Galluppi, Steve Temple, and Luis A. Plana (2014). "The SpiNNaker Project". In: *Proceedings of the IEEE* 102.5, pp. 652–665. ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2304638 (cit. on pp. 2, 23).

Gerstner, Wulfram and Werner M. Kistler (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press. ISBN: 9780521890793 (cit. on p. 5).

Gewaltig, Marc-Oliver and Markus Diesmann (2007). "NEST (NEural Simulation Tool)". In: *Scholarpedia* 2.4, p. 1430. DOI: 10.4249/scholarpedia.1430 (cit. on p. 142).

Gorel, Alexander (2013). "Integration einer automatisierten analogen und erweiterbaren Testumgebung zur Validierung und Überwachung von Hardware und Software Frameworks". In: (cit. on p. 30).

Graves, Alex, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. (2016). "Hybrid computing using a neural network with dynamic external memory". In: *Nature* 538.7626, pp. 471–476. DOI: 10.1109/IJCNN.2006.246651 (cit. on p. 1).

Hartel, Andreas (2016). "Implementation and Characterization of Mixed-Signal Neuromorphic ASICs". PhD thesis. Universität Heidelberg (cit. on pp. 10, 60–61, 145).

Hasler, Jennifer and Harry Bo Marr (2013). "Finding a Roadmap to achieve Large Neuromorphic Hardware Systems". In: *Frontiers in Neuroscience* 7.118. ISSN: 1662-453X. DOI: 10.3389/fnins.2013.00118 (cit. on p. 2).

Hasler, Paul, Bradley A. Minch, and Chris Diorio (1999). "Floating-gate devices: they are not just for digital memories any more". In: *Circuits and Systems, 1999.*

ISCAS'99. Proceedings of the 1999 IEEE International Symposium on. Vol. 2. IEEE, pp. 388–391. DOI: 10.1109/ISCAS.1999.780740 (cit. on p. 16).

Hock, Matthias (2009). "Test of Components for a Wafer-Scale Neuromorphic Hardware System". Diploma thesis. Universität Heidelberg (cit. on pp. 17, 61).

– (2014). "Modern semiconductor technologies for neuromorphic hardware". PhD thesis. Universität Heidelberg (cit. on p. 145).

Hunter, John D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cit. on pp. 28, 159).

Indiveri, Giacomo, Federico Corradi, and Ning Qiao (2015). "Neuromorphic architectures for spiking deep neural networks". In: *2015 IEEE International Electron Devices Meeting (IEDM)*. IEEE, pp. 4–2. DOI: 10.1109/IEDM.2015.7409623 (cit. on p. 1).

Indiveri, Giacomo, Bernabe Linares-Barranco, Tara Hamilton, André van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, Johannes Schemmel, Gert Cauwenberghs, John Arthur, Kai Hynna, Fopefolu Folowosele, Sylvain SAÏGHI, Teresa Serrano-Gotarredona, Jayawan Wijekoon, Yingxue Wang, and Kwabena Boahen (2011). "Neuromorphic Silicon Neuron Circuits". In: *Frontiers in Neuroscience* 5, p. 73. ISSN: 1662-453X. DOI: 10.3389/fnins.2011.00073 (cit. on p. 3).

Jeltsch, Sebastian (2014). "A Scalable Workflow for a Configurable Neuromorphic Platform". PhD thesis. Universität Heidelberg (cit. on pp. 15, 21, 23, 25–27).

Jones, Eric, Travis Oliphant, Pearu Peterson, et al. (2001–). *SciPy: Open source scientific tools for Python*. [Online; accessed 2016-11-03]. URL: http://www.scipy.org/ (cit. on p. 28).

Jupyter, Project (2015–). *Jupyter*. [Online; accessed 2016-11-03]. URL: http://www.jupyter.org/ (cit. on p. 28).

Karasenko, Vitali (2014). "A communication infrastructure for a neuromorphic system". Master thesis. Universität Heidelberg (cit. on pp. 33, 45).

Kiene, Gerd (2014). "Evaluating the Synaptic Input of a Neuromorphic Circuit". Bachelor thesis. Universität Heidelberg (cit. on pp. 15, 81–85, 88).

Klähn, Johann (2013). "Untersuchung und Management von Synapsendefektverteilungen in einem großskaligen neuromorphen Hardwaresystem". Bachelor thesis. Universität Heidelberg (cit. on p. 27).

Kononov, Alexander (2011). "Testing of an Analog Neuromorphic Network Chip". Diploma thesis. Universität Heidelberg (cit. on pp. 10, 42, 61, 64).

Kungle, Ákos Ferenc (2016). "Sampling with Leaky Integrate and Fire neurons on the HICANNv4 neuromorphic chip". Master thesis. Universität Heidelberg (cit. on p. 131).

Kunkel, Susanne, Maximilian Schmidt, Jochen M. Eppler, Hans E. Plesser, Gen Masumoto, Jun Igarashi, Shin Ishii, Tomoki Fukai, Abigail Morrison, Markus Diesmann, et al. (2014). "Spiking network simulation code for petascale computers". In: *Frontiers in neuroinformatics* 8. DOI: http://dx.doi.org/10.3389/fninf.2014.00078 (cit. on p. 1).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539 (cit. on p. 1).

LeCun, Yann, Corinna Cortes, and Christopher JC Burges (1998). *The MNIST database of handwritten digits* (cit. on p. 145).

Lenzlinger, Martin and E.H. Snow (1969). "Fowler-Nordheim Tunneling into Thermally Grown SiO2". In: *Journal of Applied physics* 40.1, pp. 278–283. DOI: 10.1063/1.1657043 (cit. on p. 17).

Loock, Jan-Peter (2006). "Evaluierung eines Floating Gate Analogspeichers für Neuronale Netze in Single-Poly UMC 180nm CMOS-Prozess". Diploma thesis. Universität Heidelberg (cit. on pp. 16, 18, 63).

Lovett, Simon J., Marco Welten, Alan Mathewson, and Barry Mason (1998). "Optimizing MOS transistor mismatch". In: *IEEE Journal of Solid-State Circuits* 33.1, pp. 147–150. DOI: 10.1109/4.654947 (cit. on p. 41).

Markram, Henry, Eilif Muller, Srikanth Ramaswamy, Michael W. Reimann, Marwan Abdellah, Carlos Aguado Sanchez, Anastasia Ailamaki, Lidia Alonso-Nanclares, Nicolas Antille, Selim Arsever, et al. (2015). "Reconstruction and simulation of neocortical microcircuitry". In: *Cell* 163.2, pp. 456–492. DOI: 10.1016/j.cell.2015.09.029 (cit. on p. 1).

Mauch, Christian (2016). "Commissioning of a Neuromorphic Computing Platform". Master thesis. Universität Heidelberg (cit. on pp. 21, 48, 140).

McKinney, Wes (2013). *Python for data analysis. [agile tools for real-world data]*. eng. 1. ed., 2. release. O'Reilly, XIII, 452 S. ISBN: 978-1-4493-1979-3 (cit. on p. 28).

Merolla, Paul A., John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha (2014). "A million spiking-neuron integrated circuit with a scalable communication network and interface". In: *Science* 345.6197, pp. 668–673. DOI: 10.1126/science.1254642 (cit. on p. 2).

Millner, Sebastian (2012). "Development of a Multi-Compartment Neuron Model Emulation". PhD thesis. Universität Heidelberg (cit. on pp. 5, 10, 16–19, 22, 42, 46, 56, 81, 85–86, 89, 91, 107, 109, 134, 156).

Millner, Sebastian, Andreas Grübl, Karlheinz Meier, Johannes Schemmel, and Marc-Olivier Schwartz (2010). "A VLSI Implementation of the Adaptive Exponential Integrate-and-Fire Neuron Model". In: *Advances in Neural Information Processing Systems* 23, pp. 1642–1650. URL: http://papers.nips.cc/paper/3995-a-vlsi-implementation-of-the-adaptive-exponential-integrate-and-fire-neuron-model.pdf (cit. on p. 10).

Müller, Eric C. (2014). "Novel Operation Modes of Accelerated Neuromorphic Hardware". PhD thesis. Universität Heidelberg (cit. on pp. 19, 21–23, 27, 31, 33, 45, 55).

Naud, Richard, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner (2008). "Firing patterns in the adaptive exponential integrate-and-fire model". In: *Biolog-*

ical cybernetics 99.4-5, pp. 335–347. DOI: 10.1007/s00422-008-0264-7 (cit. on pp. 5–7).

SP9 Spec (2015). *Neuromorphic Platform Specification*. Human Brain Project (cit. on pp. 5, 15, 18–19, 23, 37–38, 112, 153).

Nonnenmacher, Tobias (2015). "Characterization of Spike-Timing Dependent Plasticity in Neuromorphic Hardware". Master thesis. Universität Heidelberg (cit. on pp. 16, 105).

Pape, Constantin (2013). "Vergleich der Executable System Specification mit neuromorpher Hardware über eine gemeinsame Bedienungsschnittstelle". Bachelor thesis. Universität Heidelberg (cit. on p. 27).

Passig, Kathrin and Johannes Jander (2013). *Weniger schlecht programmieren*. O'Reilly Verlag. ISBN: 9783897215689 (cit. on p. 32).

Pelgrom, Marcel J.M., Aad C.J. Duinmaijer, Anton P.G. Welbers, et al. (1989). "Matching properties of MOS transistors". In: *IEEE Journal of solid-state circuits* 24.5, pp. 1433–1439. DOI: 10.1016/0168-9002(91)90167-0 (cit. on pp. 3, 41, 78).

Pérez, Fernando and Brian E. Granger (2007). "IPython: a System for Interactive Scientific Computing". In: *Computing in Science and Engineering* 9.3, pp. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53 (cit. on p. 28).

Petrovici, Mihai A. (2016). *Form Versus Function: Theory and Models for Neuronal Substrates*. Springer. ISBN: 978-3-319-39552-4 (cit. on p. 8).

Pfeil, Thomas, Andreas Grübl, Sebastian Jeltsch, Eric Müller, Paul Müller, Mihai A. Petrovici, Michael Schmuker, Daniel Brüderle, Johannes Schemmel, and Karlheinz Meier (2013). "Six networks on a universal neuromorphic computing substrate". In: *Frontiers in Neuroscience* 7, p. 11. DOI: 10.3389/fnins.2013.00011 (cit. on p. 3).

Pfeil, Thomas, Anne-Christine Scherzer, Johannes Schemmel, and Karlheinz Meier (2013). "Neuromorphic learning towards nano second precision". In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, pp. 1–5. DOI: 10.1109/IJCNN.2013.6706828 (cit. on p. 3).

Pilz, Lukas (2016). "Towards Fast Iterative Learning On The BrainScaleS Neuromorphic Hardware System". Bachelor thesis. Universität Heidelberg (cit. on pp. 21, 44).

Quarles, Thomas L. (1989). "Analysis of Performance and Convergence Issues for Circuit Simulation". PhD thesis. EECS Department, University of California, Berkeley. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1989/1216.html (cit. on p. 53).

Razavi, Behzad (2001). *Design of analog CMOS integrated circuits*. eng. International ed. McGraw-Hill series in electrical and computer engineering ; Electrical engineering series. Hier auch später erschienene, unveränderte Nachdrucke. Boston, Mass. [u.a.]: McGraw-Hill, p. 684. ISBN: 0-07-118839-8 (cit. on pp. 3, 41, 82).

Rudolph, Michelle and Alain Destexhe (2006). "Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies". In: *Neural computation* 18.9, pp. 2146–2210. DOI: 10.1162/neco.2006.18.9.2146 (cit. on p. 8).

Schemmel, Johannes (2015). "Deliverable 3-1.2 Annex 1 – The BrainScaleS Neuromorphic Hardware System A BrainScaleS Internal Document." In: *BrainScaleS internal documentation* (cit. on p. 107).

Schemmel, Johannes, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner (2010). "A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neural Modeling". In: *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS"10)*, pp. 1947–1950. DOI: 10.1109/ISCAS.2010.5536970 (cit. on pp. 3, 15, 21).

Schemmel, Johannes, Daniel Brüderle, Karlheinz Meier, and Boris Ostendorf (2007). "Modeling Synaptic Plasticity within Networks of Highly Accelerated I&F Neurons". In: *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE Press, pp. 3367–3370. DOI: 10.1109/ISCAS.2007.378289 (cit. on p. 5).

Schemmel, Johannes, Johannes Fieres, and Karlheinz Meier (2008). "Wafer-scale integration of analog neural networks". In: *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, pp. 431–438. DOI: 10.1109/IJCNN.2008.4633828 (cit. on pp. 5, 15, 45).

Schemmel, Johannes, Andreas Grübl, Stephan Hartmann, Alexander Kononov, Christian Mayr, Karlheinz Meier, Sebastian Millner, Johannes Partzsch, Stefan Schiefer, Stefan Scholze, René Schuffny, and Marc-Oliver Schwartz (2012). "Live demonstration: A scaled-down version of the BrainScaleS wafer-scale neuromorphic system". In: p. 702. DOI: 10.1109/ISCAS.2012.6272131 (cit. on p. 22).

Schemmel, Johannes, Andreas Grubl, Karlheinz Meier, and Eilif Mueller (2006). "Implementing synaptic plasticity in a VLSI spiking neural network model". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, pp. 1–6. DOI: 10.1109/IJCNN.2006.246651 (cit. on p. 3).

Schmidt, Dominik (2014). "Automated Characterization of a Wafer-Scale Neuromorphic Hardware System". Master thesis. Universität Heidelberg (cit. on pp. 12, 15, 19, 22, 43, 66–67, 99–100, 104–105, 134).

Schmitt, Sebastian, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Güttler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch, Vitali Karasenko, Mitja Kleider, Christoph Koke, Alexander Kononov, Christian Mauch, Eric Müller, Paul Müller, Johannes Partzsch, Mihai A. Petrovici, Bernhard Vogginger, Stefan Schiefer, Stefan Scholze, Vasilis Thanasoulis, Johannes Schemmel, Robert Legenstein, Wolfgang Maass, Christian Mayr, and Karlheinz Meier (2017). *Classification With Deep Neural Networks on an Accelerated Analog Neuromorphic System*. submitted to ICJNN 2017 (cit. on p. 145).

Scholze, Stefan, Holger Eisenreich, Sebastian Höppner, Georg Ellguth, Stephan Henker, Mario Ander, Stefan Hänzsche, Johannes Partzsch, Christian Mayr, and René Schüffny (2012). "A 32GBit/s communication SoC for a waferscale neuromorphic system". In: *INTEGRATION, the VLSI journal* 45.1, pp. 61–75. DOI: 10.1016/j.vlsi.2011.05.003 (cit. on p. 45).

Scholze, Stefan, Stefan Schiefer, Johannes Partzsch, Stephan Hartmann, Christian Georg Mayr, Sebastian Höppner, Holger Eisenreich, Stephan Henker, Bernhard

Vogginger, and Rene Schüffny (2011). "VLSI implementation of a 2.8 Gevent/s packet-based AER interface with routing and event sorting functionality". In: *Frontiers in neuroscience* 5. DOI: 10.3389/fnins.2011.00117 (cit. on p. 21).

Schwartz, Marc-Olivier (2013). "Reproducing Biologically Realistic Regimes on a Highly-Accelerated Neuromorphic Hardware System". PhD thesis. Universität Heidelberg (cit. on pp. 3, 22, 41, 43, 66–67, 81, 104, 134).

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587, pp. 484–489. DOI: 10.1038/nature16961 (cit. on p. 1).

Silver, Rae, Kwabena Boahen, Sten Grillner, Nancy Kopell, and Kathie L. Olsen (2007). "Neurotech for neuroscience: unifying concepts, organizing principles, and emerging tools". In: *The Journal of Neuroscience* 27.44, pp. 11807–11819. DOI: 10.1523/JNEUROSCI.3575-07.2007 (cit. on p. 3).

Srowig, André, Jan-Peter Loock, Karlheinz Meier, Johannes Schemmel, Holger Eisenreich, Georg Ellguth, and René Schüffny (2007). "Analog Floating Gate Memory in a 0.18 μm Single-Poly CMOS Process". In: *FACETS internal documentation* (cit. on pp. 16–17).

Stroustrup, B. (2013). *The C++ Programming Language*. Pearson Education. ISBN: 9780133522853 (cit. on p. 159).

Sutter, Herb and Andrei Alexandrescu (2004). *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices*. C++ In-Depth Series. Pearson Education. ISBN: 9780132654425. URL: https://books.google.de/books?id=mmjVIC6WolgC (cit. on p. 26).

Thanasoulis, Vasilis, Bernhard Vogginger, Johannes Partzsch, and René Schuffny (2014). "A pulse communication flow ready for accelerated neuromorphic experiments". In: *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pp. 265–268. DOI: 10.1109/ISCAS.2014.6865116 (cit. on pp. 21, 33).

TOP 500 (2015). *EPFL Blue Brain IV - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect*. URL: http://www.top500.org/system/178198 (visited on 01/19/2015) (cit. on p. 1).

Tran, Binh (2013). "Demonstrationsexperimente auf neuromorpher Hardware". Bachelor thesis. Universität Heidelberg (cit. on pp. 7, 29).

UMC (2004). *Release Notes for UMC 0.18um RFCMOS RF MODEL LIBRARY Release: Ver 2d3*. [The document is subjected to an NDA signed by the author.] (cit. on p. 53).

Vogginger, Bernhard (2010). "Testing the Operation Workflow of a Neuromorphic Hardware System with a Functionally Accurate Model". Diploma thesis. Universität Heidelberg (cit. on p. 25).

Weste, N. and D. Harris (2011). *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson Education. ISBN: 9780133001471 (cit. on p. 41).

Zenke, Friedemann and Wulfram Gerstner (2014). "Limits to high-speed simulations of spiking neural networks using general-purpose computers". In: *Frontiers in*

Neuroinformatics 8, p. 76. ISSN: 1662-5196. DOI: 10.3389/fninf.2014.00076 (cit. on p. 1).
Ziegler, Simon (2013). "Optimierung der physikalischen Signalübertragung auf neuromorpher Hardware". Bachelor thesis. Universität Heidelberg (cit. on pp. 15, 45).

Acknowledgments

I would like to thank

- Prof. Karlheinz Meier for supporting me in writing this thesis;
- Prof. Michael Hausmann to bring up the time to be a referee for this thesis as well as Prof. Ulrich Schwarz and Prof. Peter Fischer to be part of the defense committee;
- European Union for founding our work as part of BrainScaleS as well as of HBP;
- Johannes Schemmel and all colleagues from the Electronic Visions Group;
- the open software projects we used;
- my wife Anne Alice Koke of proof reading, corrections, graphics and support;
- for proof-reading and suggestions: Johannes Bill, Sebastian Billaudelle, Ilja Bytschok, Maurice Güttler, Andi Hartel, Gerd Kiene, Mitja Kleider, Christian Mauch, Stephan Meister, Eric Müller, Paul Müller, Sebastian Schmitt, and Julian Stürmer; and
- my offices mates for having a good time: first Tom Pfeil and Sebastian Jeltsch and later Ilja Bytschok and Mitja Kleider.

Lastly, I would like to express my gratitude to all who contributed to the BrainScaleS System. These are

- the designers of HICANN: Johannes Fieres, Simon Friedmann, Andreas Grübel, Andreas Hartel, Sebastian Millner, Stefan Philipp and Johannes Schemmel;
- the designers and maintainers of the wafer module components: Dan Husmann, Maurice Güttler, Joscha Ilmberger and Lars Sterzenbach;
- the designers and maintainers of the FCP: Johannes Partzsch, Stephan Scholze, Stephan Schiefer, Stephan Hartmann, Vasillis Thanasoulis and Vitali Karasenko;
- the designers and maintainers of AnaRM: Andreas Grübel, Andreas Hartel, Matthias Hock and Johannes Schemmel; and
- all, who wrote software for the BrainScaleS System: Dennis Alevi, Sebastian Billaudelle, Oliver Breitwieser, Matthias Ehrlich, Alex Gorel, Stefan Hartmann, David Hinrichs, Kai Husmann, Sebastian Jeltsch, Johann Klähn, Mitja Kleider,

Ioannis Kokkinos, Alexander Kononov, Ákos Kungl, Christian Mauch, Eric Müller, Paul Müller, Moritz Schilling, Dominik Schmidt, Sebastian Schmitt, Sven Schrader, Binh Tran, Constantin Pape, Johannes Partzsch, Lukas Pilz, Bernhard Vogginger Karsten Wendt and Simon Ziegler.

I compiled these list of contributes to the best of my knowledge and belief, to anybody who might got lost my sincere apologies.

Statement of Originality (Erklärung)

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

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

Heidelberg, den 2. Januar 2017

.....
(signature)