Department of Physics and Astronomy

University of Heidelberg

Master Thesis

in Physics

submitted by

Aron Leibfried

born in Bad Friedrichshall, Germany

April 2021

On-chip calibration and closed-loop experiments on analog neuromorphic hardware

This Master Thesis has been carried out by

Aron Leibfried

at the

Kirchhoff-Institute for Physics

Heidelberg University

under the supervision of

Dr. Johannes Schemmel

Abstract

Analog neuromorphic circuits – as other analog designs – suffer from fixed pattern noise, i.e. deviations in the behaviour of different instances of otherwise identical circuits. Calibration can facilitate experiments on such devices, by equalizing their dynamics. In the framework of this thesis we present calibration algorithms for the BrainScaleS-2 system. In particular we make extensive use of the embedded plasticity processing unit (PPU) and its vector unit to accelerate the calibration routines and guarantee their scalability on multi scale systems: the dynamics of all neurons on a chip can be calibrated on the order of seconds.

These calibration routines lay the foundation for porting a closed-loop experiment to the latest generation of BrainScaleS chips, again fully controlled by on-chip processors. It implements a neuromorphic agent steered by a neural model of insect navigation. By performing path integration, the virtual insect is able to return to its nest after a foraging trip. While the neural dynamics were emulated on the analog neuromorphic circuit, the agent's environment, the sensors, and motor units were simulated by the on-chip PPU.

Zusammenfassung

Analoge neuromorphe Schaltungen weisen aufgrund von Fertigungstoleranzen Abweichungen im Verhalten verschieder Chipinstanzen auf. Eine Kalibration kann Experimente auf solchen Geräten erleichtern, indem sie deren Dynamik angleicht. Im Rahmen dieser Arbeit werden Kalibrationsmechanismen für das BrainScaleS-2 System vorgestellt. Insbesondere wird dafür der auf dem Chip implementierte Prozessor (PPU) und die dazugehörige Vektoreinheit verwendet, um schnelle Kalibrationsroutinen zu ermöglichen und ihre Skalierbarkeit bezogen auf Systeme mit mehreren Chips zu gewährleisten. Die Dynamik aller Neuronen auf einem Chip kann innerhalb von Sekunden kalibriert werden.

Diese Kalibrationsroutinen legen den Grundstein für die Portierung eines Closed-loop Experiments auf die neueste Generation der BrainScaleS Chips, welches wiederum selbst vollständig vom implementierten Prozessor gesteuert wird. Dieses Experiment implementiert ein neuronales Netz, welches die Navigation von Insekten emuliert. Durch Pfadintegration ist das virtuelle Insekt in der Lage nach seiner Futtersuche zu seinem Nest zurückzukehren. Während die neuronale Dynamik auf dem analogen neuromorphen Schaltkreis emuliert wurde, wurden die Umgebung, die Sensoren und die Motorsignale vom eingebetteten Prozessor simuliert.

Contents

1. Introduction

Traditional computers based on the von Neumann architecture play a key role in modern science. Supercomputers like the *Fugaku* are able to perform $488 \cdot 10^{15}$ floating-point operations per second [Kodama et al., 2020]. This computational power is used in various scientific fields, for example in cosmological simulations of galaxy formations [Vogelsberger et al., 2019] or for processing big data in the ATLAS experiment at CERN [Borodin et al., 2015].

With these traditional computers it is also possible to realize artificial neural networks (ANN). Especially when trained with gradient descent in deep multi-layer networks, they can be used in many applications [LeCun et al., 2015]. Besides already widely used tasks like image classification [Krizhevsky et al., 2017] or text translation [Singh et al., 2017] even the realization of self-driving cars is under development [Rao and Frtunikj, 2018]. Using more complex models involving more hidden layers was the recent strategy for better performance [Bianchini and Scarselli, 2014].

Von Neumann computers are designed to perform arithmetic operations with fast calculation speed and a low susceptibility to errors. The human brain however is very creative and flexible, it forms instinct and solves a wide variety of tasks based on experience and logical conclusions. Artificial neural networks are always bound to the tasks they were designed for. Besides its flexibility, the human brain only requires a few watts to perform its task, while supercomputers require power on the order of megawatts [Mehmood et al., 2018], making the human brain way more energy efficient for learning tasks. Based on discrete spikes to communicate and transmit information, spiking neural networks (SNN) are more biologically plausible to describe the processes in the brain [Tavanaei et al., 2019].

Until recently, the development of traditional computers proceeded at an exponential pace, as expressed by Moore's law, making it possible to execute more calculations in less time. A wide consensus exists about the ending of this law, so developing novel principles, like brain-inspired computing [Peper, 2017], is required. Also the brain's low energy consumption and flexibility are good reasons for attempting to mimic the brain. A wide variety of neuromorphic systems specialized for the replication of neural networks exists, like Intel's Loihi [Davies et al., 2018], IBM's TrueNorth [DeBole et al., 2019] and SpiNNaker [Furber et al., 2014].

In this thesis, HICANN-X [Schemmel et al., 2020], the latest iteration of the BrainScaleS-2 family, was used. The system contains analog and digital parts and was primarily designed to emulate spiking neural networks, but it can also perform vector-matrix multiplication used in artificial neural networks [Weis et al., 2020].

One important focus of this thesis is the presentation of new calibration routines for the analog circuitry. Calibration is necessary for two reasons: Analog circuits exhibit fixed-pattern deviations between multiple instances of the same design due to variations in the production process. When supported by the design, these effects can be equilibriated by suitable configuration routines. In this process, a specific operating point defined by a set of high-level parameters such as time constants can be targeted. A calibration routine has to reliably reach the desired targets and do so with as little overhead as possible. In particular this can be challenging when considering distributed systems consisting of dozens or even hundreds of individual ASICs. In this case, a parallel approach was developed using the on-chip digital

processor. Calibrations for the BrainScaleS-2 family already exist, but compared to the parallel approach they do not scale with the amount of ASICs. For this reason, the runtime of the presented calibrations should stay approximately constant even for larger systems.

We furthermore test our on-chip calibration routines in biological inspired closed-loop experiments. Some models implementing parts of insect brains get along with just a few neurons [Stone et al., 2017]. In this thesis such a small-scaled network is presented implementing a neuromorphic nervous system. This neural network allows an artificial bee to navigate through a virtual environment finding its way back to its nest after the search for food. Only 18 neurons are used for the implementation on the analog part of the chip. The other information necessary to perform this experiment was simulated on the on-chip processor, including the generation of input signals, processing of the motor signals and simulating the virtual environment.

2. HICANN-X: From biology to silicon

The ability to form instinctive and intelligent behavior is made possible by an estimated 100 billion neurons [Herculano-Houzel, 2012] and 150 trillion synapses [Pakkenberg et al., 2003] in the human brain. According to a general picture, a neuron consists of three different parts. The dendrites branch out from the soma and form synapses with other neurons. This allows the soma to pick up signals from different synaptic partners. When a critical membrane potential is reached, an action potential is sent along the axon, where synaptic contacts with other neurons' dendrites are activated [Eyzaguirre and Kuffler, 1955].

HICANN-X is the latest iteration of the BrainScaleS-2 system, designed to emulate synapses, neurons and plasticity models in spiking neural networks [Schemmel et al., 2020]. The aim is to provide an experimental platform to emulate the brain's components to study the dynamics of the brain. In contrast to other systems like SpiNNaker [Mayr et al., 2019], the neurons and synapses are based on analog circuits, manufactured in standard 65 nm CMOS technology. Due to this implementation, biologically inspired networks are emulated in 1000-fold accelerated time on BrainScaleS-2. Within this thesis all times and parameters are given in the real chip time domain, not as they would appear in biology.

The chip features 512 silicon neurons receiving input signals from 256 individual synapses each, so the total number of synapses is 131 072. The neurons are arranged in two rows with two synapse matrices as one can also conclude from the left picture in figure 2.1 showing a photograph of the ASIC. Every row is again split into two halves, resulting in four quadrants of 128 neurons each.

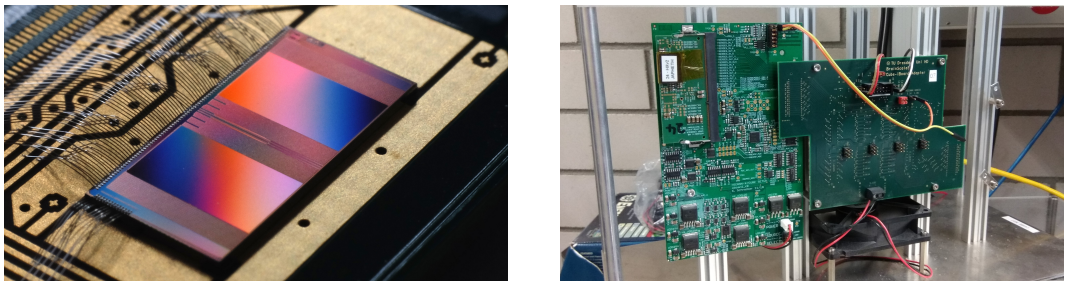

Figure 2.1: Photograph of the bonded HICANN-X ASIC (left) and the hardware setup (right). The hardware setup includes power and communication boards with the actual chip being bonded on a SO-DIMM module and covered by a black cap with a white label.

The right side of figure 2.1 shows the hardware setup used in this thesis, providing boards for the power supply as well as test connections. It also features a field programmable gate array (FPGA) that handles the instructions sent by a host computer via Ethernet to control the HICANN-X chip in real time.

Besides the FPGA also the two on chip microprocessors (plasticity processing unit, PPU), one for each synapse matrix, are able to control experiments autonomously. Most work presented in this thesis was done from the PPUs.

2.1. The neuron circuit

The neuron circuit on HICANN-X emulates the adaptive exponential integrate and fire model (AdEx) [Brette and Gerstner, 2005], an extension to the classical leaky-integrate and fire model (LIF). It allows for more realistic action potential shapes and firing patterns as they appear in biology [Naud et al., 2008]. For this thesis only the parts from the LIF model are relevant, all AdEx extensions were disabled. The basic circuitry is shown in figure 2.2. In this model the neuron's membrane is realized as a capacitor. An operational transconductance amplifier (OTA) implements the leak by acting as a pseudo-resistor pulling the membrane to the leak potential V_{leak}. Implementing a current-based model [Cavallari et al., 2014], the synaptic inputs are also realized by OTAs. When a specified threshold potential V_{thresh} is reached, the membrane is pulled to a reset potential V_{reset}. This event is triggered by the spike threshold comparator, completing the LIF neuron. How long the neuron is pulled to the reset potential is specified by the refractory time τ_{refr}. The circuitry is explained in more detail now.

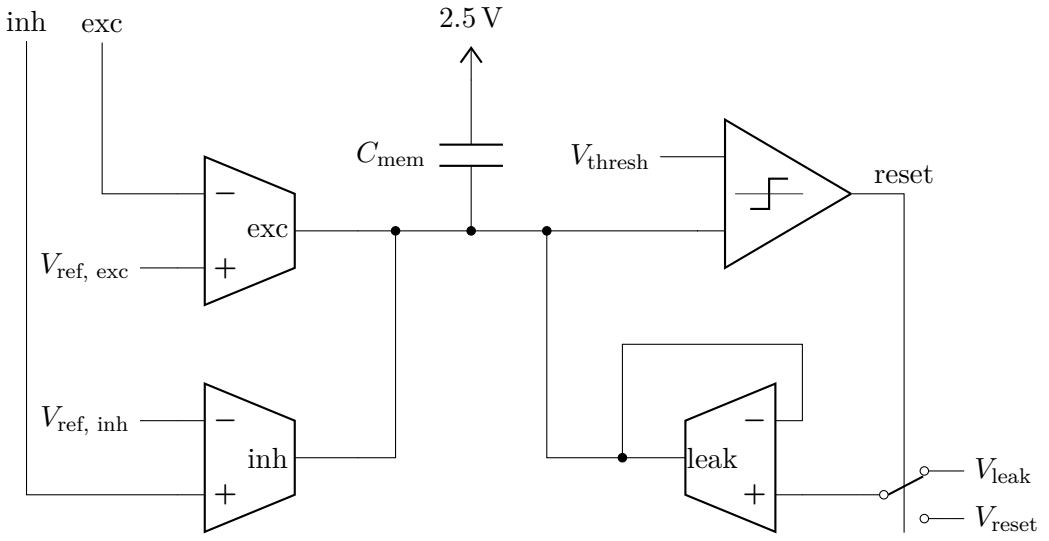

Figure 2.2: Simplified schematic of an analog LIF neuron. The membrane capacitor is the heart of the circuit receiving excitatory and inhibitory synaptic inputs via OTAs shown on the left. The synaptic lines called "exc" and "inh" get signals from the synapses. The leak OTA on the right pulls the membrane back to a leak potential. If a threshold is reached the comparator on the top right emits a spike and the membrane is pulled back to the reset potential for an adjustable refractory period.

The leak OTA on the right generates a current onto the membrane which depends on the transconductance value g_{leak} and the difference between the membrane potential and V_{leak}. It can be seen as a tunable resistor-like element. After a spike occurred, the neuron is in the refractory period, which is set digitally by a counter. During this time, the leak OTA serves as neuron reset and pulls the membrane to V_{reset}. The transconductance value can be set individually for both modes to ensure a fast reset, while one also gets a higher membrane time constant τ_{mem}, which

depends on the g_{leak} and the membrane capacitance C_{mem}.

Without additional inputs, the membrane would be kept on the leak potential or, in case the leak is above the threshold, the neuron would spike regularly, which is called leak-over-threshold. To get a dynamic system with interactions between different neuron instances, the synaptic input plays an important role. There are two instances which are in principle identical and can be seen on the left side in figure 2.2. The only difference is the changed polarity of the OTA, so the excitatory input charges the membrane while the inhibitory input discharges it.

The synaptic input lines are marked with "exc" and "inh", receiving input currents from the different synapses of the according neuron. These inputs decrease the voltage on these lines by an amount $\Delta U = Q_{\text{syn}}/C$, with the charge $Q_{\text{syn}}(w)$ "sinked" by each individual synapse per event. The line's capacitance C of the synaptic input results from the parasitic capacitance C_{line} of the line and an additional detachable capacitor C_{syn}, both not shown in the figure. A tunable resistor-like element R_{syn}, also not shown, pulls the line back to the supply voltage of 1.2 V. Every voltage difference will exponentially decay back to the supply voltage with the synaptic time constant τ_{syn}, depending on the resistor-like element and the line's capacitance. If input events are absent, the reference potential should be be adjusted such that no current flows on the membrane. The resulting current onto the membrane depends on the transconductances of the OTAs and the difference between the synaptic lines potentials and the reference potentials.

2.2. Spike generation and synapses

Propagating spikes through the chip is done digitally. Spikes are generated in the synapse drivers and send to the synapses as visualized in figure 2.3. These synapse drivers are able to perform short-term plasticity (STP) [Fioravante and Regehr, 2011], based on synaptic changes resulting from prior activity of the synapses. The according model used on hardware is derived from the Tsodyks-Markram model [Tsodyks and Markram, 1997]. For this thesis STP is not of interest and therefore disabled. As a result, all current pulses send from the drivers will have a length of 4 ns and are not influenced by the STP circuitry.

Every synapse driver is connected to two synapse rows, so 128 drivers are implemented per half. Each row can be individually configured to serve either as excitatory or inhibitory input. Individual 6-bit weights w and addresses can be set for every single synapse. While the length of the current stimulus $\Delta t = 4\,\text{ns}$ onto the neuron is determined by the drivers, the height $I(w)$ is modulated in each synapse. The total emitted charge per spike $Q_{\text{syn}} = I \cdot \Delta t$ is the product of both parameters. If the programmed address of the synapse matches with the spike address, this current stimulus is sent to the neuron in the same column, able to stimulate the membrane via the synaptic inputs.

There are three sources of events generating spikes with a desired address. One can either send external spikes manually with the desired address, specify an output address for a neuron to generate an internal spike when its threshold is reached or use one of the eight different on chip background spike generators. Each background spike generator can be configured to send spikes to a desired address, with random masks also to several addresses. It can either send regular or poisson spike trains, each can be configured with a desired spiking frequency.

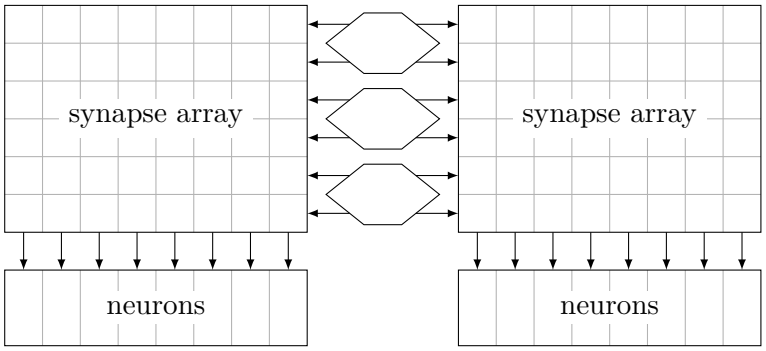

Figure 2.3: Visualisation of the upper chip-half in the analog part of HICANN-X. The lower chip-half is mirrored downwards. Signals from the synapse drivers in the middle travel in the synapse array via the synapses to the according neuron in the column. Two synapse rows are connected to one driver each.

2.3. Plasticity processing unit

The plasticity processing unit (PPU), a general-purpose 32-bit processor implementing the PowerPC-ISA 2.06 instruction set [Friedmann et al., 2017], was originally introduced to realize spike time dependent plasticity (STDP) [Bi and Poo, 2001] and other plasticity rules. It is extended with a custom 1024-bit vector extension, which can be used for calculation but also for setting hardware parameters in the synapse array like the weights of synapses. Every chip contains two PPU instances, one for the upper half of the chip and the other one for the bottom half. Like the FPGA, they are able to control experiments and set hardware parameters.

They can either be programmed in assembler or in C++. The PPUs are clocked with 250 MHz and have access to 16 kB main memory each. If more memory is needed, the PPUs can also access external memory in the FPGA to store code or data. The program binary must be loaded via the FPGA, but afterwards the PPU can control the whole chip completely independently of the host computer.

Both PPUs are able to read out the columnar analog-to-digital converters (CADCs), 8-bit ADCs. They were initially implemented to read out correlations from synapses for STDP learning rules, but they can also be used to digitize the membrane potential of each neuron at once. In addition, neuron rate counters can be read out, which is a individual spike counter for every neuron. The rate counters have 8 bit and every time a spike is registered, the value is incremented by one. A ninth bit serves as overflow bit.

3. Calibration: Executed on-chip

ASICs often suffer from certain manufacturing tolerances, which lead to mismatch between identically designed components. For example, a local decrease in dopand concentration could result in a reduced transconductance of a transistor. In digital systems this usually influences the maximum clock frequency available. In analog integrated circuitry such deviations directly influence the very properties and dynamics of the design. When it comes to neuromorphic designs, a stronger synaptic input for example could lead to more output spikes because the threshold is reached faster. To reduce this mismatch between individual neurons and other components, calibration algorithms are necessary.

The LIF neuron depends on many different parameters, so various calibration routines have to be developed. On BrainScaleS-2 for each neuron 8 voltages and 16 currents can be uniquely tuned to achieve a desired behaviour. These are generated by a capacitive memory (CapMem) based on 10 bit parameters [Hock et al., 2013]. 10 of these CapMem cells are assigned to the usage of the LIF model. In addition to the neuron-specific cells, there are also cells that are responsible for quadrant-wide or chip-wide parameters and are used, for example, in the CADC or the synapse drivers. Individual calibration of these voltages and currents in the various components makes it possible to reduce mismatch. Besides the CapMem also other component-specific calibration mechanisms are available.

The goal of this thesis was to use the PPU for calibration. This has several advantages, for example during calibration no data and commands have to be exchanged with the host computer once the program started. So the calibration is completely independent of network delays and external data inputs. This becomes especially important for multi-chip systems such as wafer-scale platforms: Since all computation is performed locally, PPU-based calibration algorithms could exhibit perfect weak scaling [Shoukourian et al., 2014].

One disadvantage of an on-chip calibration is the reduced number of observables, which are limited to the voltages measured by the CADCs and the spike counters. Additionally, the development effort is much higher and also more complex. First steps calibrating BrainScaleS-2 systems on the PPU were made on the predecessor of HICANN-X, HICANN-DLSv3 [Leibfried, 2018]. Some of these concepts were also used and extended in this thesis. An important goal was the calibration of the neuron parameters towards desired target values, which was not possible in the existing calibration, which in contrast only allowed to equilibriate the dynamics across neurons without specific targets. As far as possible, every calculation and configuration was performed on the vector unit in order to keep the computing overhead as small as possible.

3.1. Software structure

Different parameters need to be calibrated on chip, so a general functions was developed to handle the calibration algorithm. This generic routine was based on a binary search since most parameters and their influence on the observables are monotonic. The calibration should calibrate the according parameters so that the observable is very close to the given target value. Functions to set the parameter and to get the resulting observables had to be developed individually for each parameter.

An issue with the CapMem arises when many CapMem cells are set to the same digital value. In this case the generated voltage or current in a cell is different from the value obtained with only one cell active. Due to this crosstalk it is not possible to use a "standard" binary search with the same starting values for all CapMem parameters. So a `do_binary_search()` function was developed. It was overloaded to take arrays as well as single parameters. Specialized for the calibration of CapMem values a `do_noisy_binary_search()` function was developed to deal with the crosstalk problematic. Both algorithms are based on the same principle. The following pseudo code illustrates the procedure for both.

```
1 for i in [1 .. iterations]:
2     set_bit = 1 << (iterations - 1)
3     binary_add(variable, set_bit, max_value)
4     set_function(variable)
5     obs_function(observable)
6     decider = decide_function(observable, target)
7     if decider:
8         binary_sub(variable, set_bit, min_value)
```

Each routine must be given a minimum (`min_value`) and maximum (`max_value`) to define the parameter range. With these values it is calculated how many iterations (`iterations`) are necessary to cover the whole parameter range. Both routines start with the minimum value adding the most significant bit (MSB) to cover the whole range. For the noisy search a tunable amount of LSBs can be jittered during the first iteration to counteract the crosstalk problem. Afterwards the same values are added for all neurons because the jitter is kept for these values. In this thesis, always 4 bits are jittered at the start leaving a starting range of ± 8 LSB. On-chip random registers in the vector unit are used for jittering these values. These random numbers are generated by different XOR-shifts.

Since it is possible to specify a desired range for the binary search, mostly one does not have a power of two as sweeping range. So by adding a value which is too high could result in a parameter outside of the intended range. Using a saturation at the maximum value however could lead to many parameters being set to the same value, making the crosstalk problematic appear. Also the jitter for the noisy search would be lost in such scenario. Whenever a new bit is added, it is checked if the parameter is above the maximum value, which is done by the `binary_add()` function. If that is the case, the overlap is subtracted from the maximum being the new parameter for the search. With this method the jitter of the noisy search is kept. The basic principle is illustrated by this pseudo code, the implementation on the vector unit is further explained in appendix A.1.

```
1 variable += set_bit
2 if variable > max_value:
3     variable -= variable - max_value
```

For the subtraction, in a few cases, a value below the minimum value can occur due to the implemented addition. To prevent such cases, the difference from the minimum is added like it is done for the addition if the parameter is below the minimum. A `binary_sub()` takes care of this illustrated by the following pseudo code. Appendix A.2 explains the implementation on the vector unit.

```
variable -= set_bit
if variable < min_value:
    variable += min_value - variable
```

An individually specified `set_function()` is responsible to set the parameters on hardware. With a given `obs_function()` the observable influenced by the parameter is obtained. This observable is used and individually compared to a desired target value via a given `decide_function()`. According to this function the added bit is kept or subtracted.

For the noisy search an additional small binary search at the end is required, because it could happen that some values cannot reach the top end of the parameters range. The number of jittered bits marks the amount of extra iterations. So calibrating a 10-bit CapMem value with the noisy binary search takes a total of 14 iterations, including the 10-bit sweep in the noisy search and the remaining 4-bit sweep with a normal binary search.

Most calibrations in this thesis are for parameters generated by the CapMem. Changing its digital settings will not change the according voltage or current immediately. Each cell is updated individually by a voltage ramp, connecting a capacitor for each cell at the right time, specified by the 10-bit value. It takes multiple of these update until the potential on the capacitor reaches the desired value. For big parameter changes up to 15 updates have to be performed to reach deviations of less than 1 LSB [Hock, 2014]. With an update period of approximately 1.5 ms, a wait of 20 ms after configuring a new set of CapMem parameters is used to get stable voltages or currents before measuring new observables again. This should yield enough precision for the calibration even for the large parameter changes at the beginning of the binary search. To get a completely general calibration class this wait is not implemented directly in the binary search, digital values for example can change immediately, making waits unnecessary. For the CapMem parameters this wait is implemented at the start of each `obs_function()`, which are individual for each calibration class.

With the implemented algorithm only the same class of parameters can be calibrated at once. Some parameters however need to be recalibrated after changing another parameter, which is possible by implementing nested calibrations in the according observable function. For example, after changing the bias current of the synaptic input OTA one has to recalibrate the according reference voltage, otherwise the result will be distorted. Also several calibration mechanism depend on former calibrated parameters, otherwise they would not be possible on the PPU. For example, it is not possible to only calibrate the synaptic time constant, different parameters like the leak or the OTA's bias current have to be calibrated beforehand.

3.2. Calibration of the CADC

Before attempting to read out the membrane potential or the correlation for the synapses it is important to calibrate the CADCs [Schreiber, 2021]. A calibration for the CADCs was already implemented by Weis [2020], where it was performed on the host computer. The basic principles are the same. However, while in this thesis an external DAC was used for the reference voltages, in the host based calibration these are generated by a global CapMem cell.

To digitize analog voltages the CADCs use a common voltage ramp, which is individual for every quadrant, and a comparator for each of the 256 channels. The ramp starts at a tunable voltage and will rise linearly with an adjustable slope. Each channel has an own 8-bit counter starting when the ramp starts rising. A comparator in each channel stops the counter if the ramp voltage reaches the observed voltage, giving the digitized value. The maximum counter value of 255 is reached after approximately 1 µs. For this reason, the starting voltage (also called offset potential) and the slope of the ramp must be adjusted to match the desired dynamic range.

The start potential of the voltage ramp and its slope, both generated by quadrant-global CapMem cells, need to be calibrated. Also due to transistor-level mismatch, the point of triggering is different for each channel, so the results are shifted by a certain offset value to compensate this effect, which has to be additionally calibrated.

It is not possible to directly read out the actual ramp offset voltage, so a different method has to be used to calibrate the according CapMem cell. By applying a lower potential as reference to the CADCs and using a yet uncalibrated ramp current, a low digitized read is expected if the ramp offset is close to the reference. This is however also the case if the ramp offset is above the reference voltage, because then the comparator will always trigger right at the beginning. To leave some headroom for individual channel offsets and preventing that the CADCs trigger when the reference voltage is above the ramp offset voltage, calibrating to a mean value of 20 LSB for all channels is considered.

For the slope of the CADC ramp a current charging a capacitor has to be calibrated. Since the start of the ramp is already calibrated, a reference voltage should now just be high enough to mark the end of the desired dynamic range. With this reference voltage, the current should yield a chosen mean value of 230 LSB by default. Like for the ramp offset, the sought value should not be set to the maximum of 255 LSB to allow for individual channel offset calibration. Both ramp parameters are calibrated with a normal binary search without the use of the vector unit, because it would make no sense to only put two values inside a vector.

After the ramps between the four quadrants are equalized, each channel offset has to be calibrated individually. The shift for each channel is determined by applying a reference voltage in the middle of the dynamic range and triggering a digitization on all channels. Considering the range of 20 LSB to 230 LSB, a target value of 125 LSB is selected. Every CADC read should give this value if the channel offsets are equalized. This calibration is not based on a binary search, just one CADC read is required and the deviation from the target gives the offset value. The calculation is completely done on the vector unit.

The dynamic range of the CADC can now be tuned by different lower and higher reference voltages generated by the external DAC. One is limited by the CADC ramp rising only until 1.2 V, while it already starts to be less linear at 1.1 V, constraining the choice for the upper reference. The lower reference voltage should not be lower than 0.05 V, otherwise a mean read of 20 LSB is not possible, because the comparators directly trigger at the start giving values close to 0 LSB. A reliable calibration of the ramp offset would not be possible anymore.

Figures 3.1 and 3.2 show the characterization of the CADC with an applied external voltage by a DAC. The mean and the standard deviation of the channels in each quadrant are plotted. For figure 3.1, the CapMem values for ramp offset and

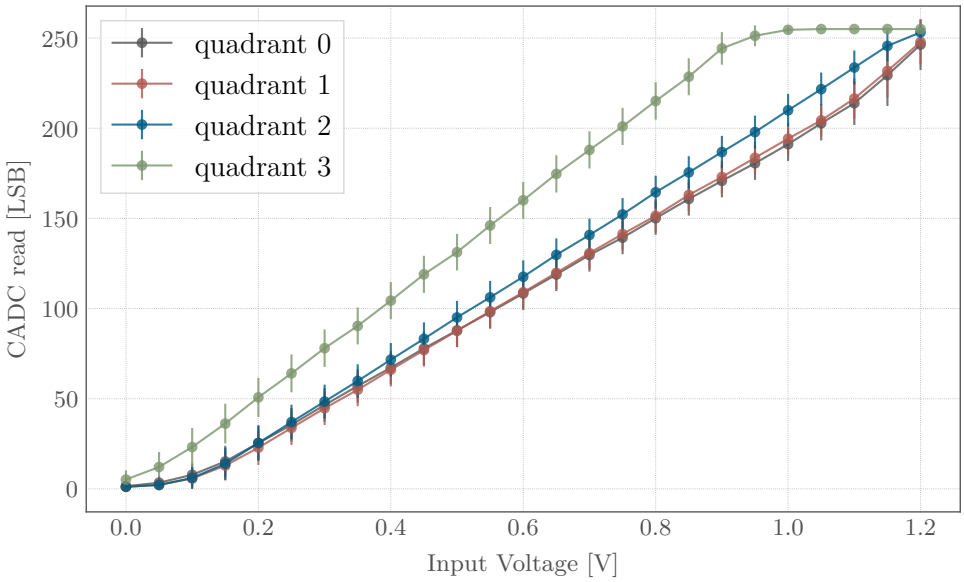

Figure 3.1: Characterization of the CADC for the uncalibrated state. An external voltage is applied and the mean and standard deviation of all 256 channels per quadrant are plotted. Different quadrants differ significantly due to the uncalibrated CADC ramp, but also the standard deviation is high showing that the individual channel offsets are not calibrated. The data was measured on setup 63 with chip 22.

slope are set to the same values for all quadrants with no individual channel offset set. It is visible that the different ramps are not aligned if uncalibrated and also the deviations between each channels are high, which is shown by the high standard deviation of each read. By using calibration, as visible in figure 3.2, the error bars are much smaller. Also the different voltages match between the different quadrants. For the plots a lower reference of 0.05 V and a upper reference of 1.1 V was chosen to get the whole dynamic range, which is sufficient for further calibrations using a calibrated CADC. The standard deviation between 0.2 V up to 1 V is always below 1 LSB. With these voltages corresponding to 40 LSB and 220 LSB one LSB corresponds to approximately 4.5 mV

3.3. Voltages

It is not possible to directly connect the output of the CapMem cells to the input of the CADCs. But as the different voltages required for the LIF neuron are influencing the membrane directly, it is possible to observe their influence on the membrane.

3.3.1. Leak voltage

With no input currents and spiking disabled, the membrane will stay at the leak potential. In this case just one CADC read delivers the observable for the calibration. The membrane time constant, in the worst case scenario around 100 µs, is not of importance as after every step in the noisy binary search a wait time of 20 ms is

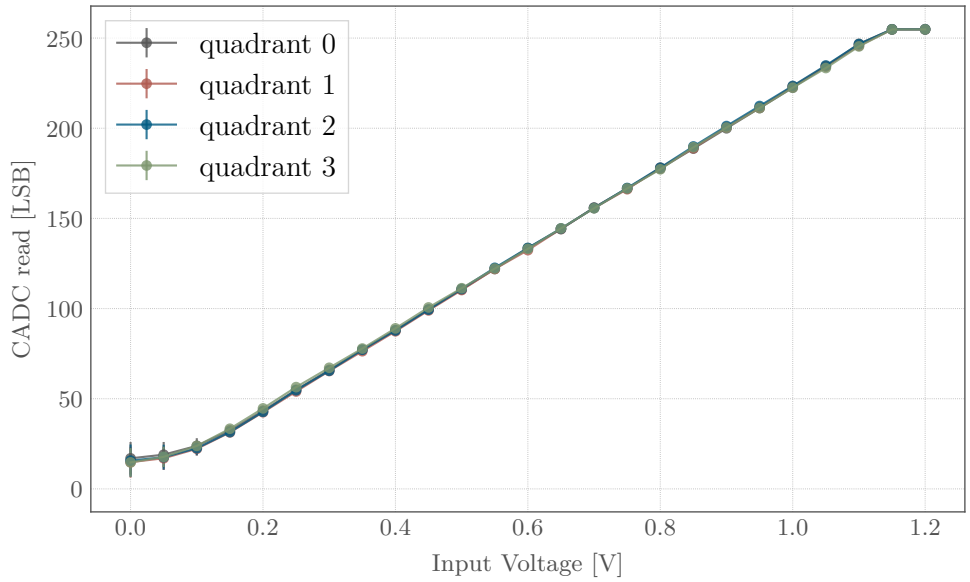

Figure 3.2: Characterization of the CADC for the calibrated state. With an external voltage applied the mean and standard deviation of all 256 channels per quadrant are determined and plotted. The characteristic of the different quadrants is now aligned and also the individual channel offsets are set that the standard deviation is below 1 LSB for input voltages from 0.2 V to 1.1 V. Outside of the dynamic range the deviations are higher due to clipping effects. The data was measured on setup 63 with chip 22.

already taken, which is at least two order of magnitudes higher.

In the current version of HICANN-X there is a source follower between CapMem cell and the actual "leak input" with a designed drop of 0.6 V. Dauer [2020] showed however that some neurons are just able to get a maximum leak potential of 0.6 V. Such voltages are all readable with the CADC.

3.3.2. Reset voltage

Neuron resets can be triggered manually via the vector unit for all neurons at once. So reading out the membrane with the CADC after such neuron reset makes is possible to calibrate it. As the CADC takes around 1 µs for reading out the membrane, it is important to have a longer refractory time such that the membrane does not start rising again during readout. Also the reset does not happen instantaneously. The membrane potential gets pulled to the reset potential by using the leak OTA with another bias value. So right after triggering a neuron reset an additional wait has to be implemented to prevent reading out a wrong value.

3.3.3. Threshold voltage

Compared to the other two voltages it is not so easy to directly read out the threshold. Reaching the threshold results in resetting the neuron, by disabling the spike reset one would just simply not know if the threshold is reached. On HICANN-

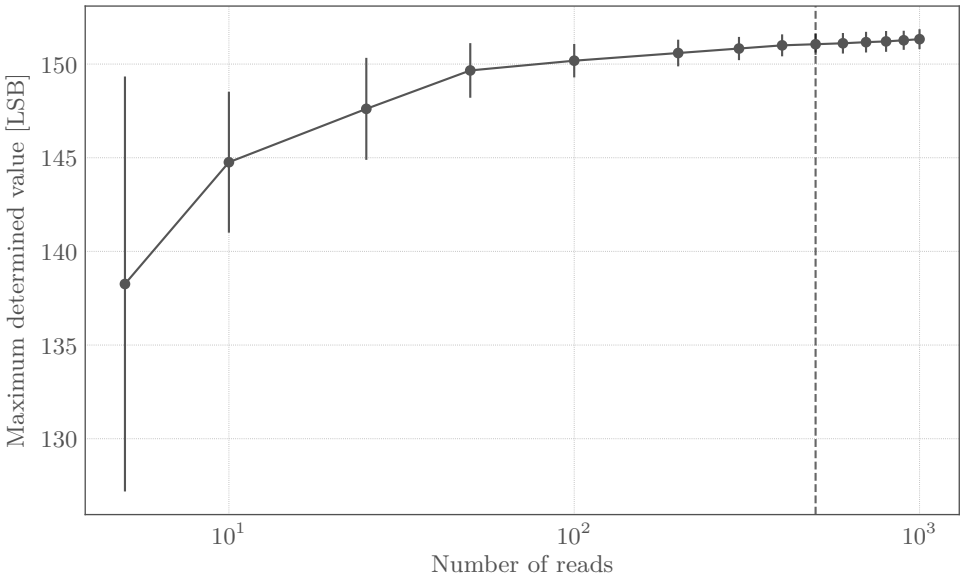

Figure 3.3: A total of 100 000 CADC reads one after the other for neuron 0 on chip 22 on setup 63 were performed with a constant current applied to the membrane, which spiked continuously. From this data 100 samples with a certain number of reads were taken and the maximum in each sample determined. Afterwards the mean and standard deviation of each determined maximum value was taken and plotted in the figure.

DLSv3 it was possible to calibrate the threshold with the help of the leak [Leibfried, 2018]. With this method after each iteration the leak was recalibrated to match the threshold. This was achieved by the spike counters, when spikes are recorded the leak is above the threshold, otherwise it is below. So the leak was brought to the point where spiking starts. With disabling spiking it was then easily possible to just read out the membrane potential with the CADC getting the threshold.

On HICANN-X this method was also implemented and worked just fine but with some drawbacks. For some neurons the leak is limited to about 0.6 V, so for them it is not possible to get a higher threshold, covering only a part of the settable range. Another problem is the nested binary search, in ever iteration step another noisy binary search was executed with another 14 iterations, which makes 196 iterations in total. This can be circumvented by first calibrating the leak to a desired level and afterwards calibrating the threshold to the point where spiking begins. In this case the problematic with the source follower still remains.

A different approach was tested to get higher threshold values. It is possible to apply a constant current to the membrane via a CapMem cell, so the neuron starts to spike regularly. By reading the membrane potential continuously with the CADCs and only taking the maximum for each neuron individually should give a good guess of the threshold. The verification is shown in figure 3.3. A total of 100 000 CADC reads of a single neuron were recorded and 100 samples taken with a different number of reads. For each sample the maximum was taken and the mean and standard deviation between the different maxima calculated and plotted.

With a small number of reads, i.e. 5 CADC reads, the maximum value cannot be determined with the CADCs. The mean lies wide away from the true value and the standard deviation is also high, so everytime a new measurement is taken different values will occur. So more reads are necessary to better estimate the threshold. For a number of 500 reads the threshold value can be closely determined within every sample, shown by a small standard deviation and also with a higher number of reads the mean value does not change. So the maximum of 500 CADC reads is taken to estimate the threshold potential in the calibration routine. The CapMem cell delivering a constant current must not be calibrated as the neuron should spike several times during these 500 CADC reads, so it is ensured to hit the target read. Unfortunately, the vector unit does not support unsigned calculations, so a method to find the maximum of an 8-bit unsigned integer on 8-bit signed arithmetic is presented in appendix A.3.

To verify the routines for calibrating different potential with the CADCs, the fast membrane-ADC (MADC) with a sampling frequency of approximately 30 MHz can be used. It can only connect to one neuron at a time and it is only possible to read it out from the host computer. Voltages are digitized to a 10 bit value, which can be translated to voltages by comparing the measured value with voltages from the external DAC. As there is a buffer between the MADC and the membrane, the voltage readout can be slightly shifted. So only differences in voltages can be reliable. To get these values, the neuron is brought to regular spiking, so it is possible to directly extract the reset and threshold potentials from the obtained curves in the MADC.

Figure 3.4 shows the difference of reset and threshold potential in an uncalibrated (top left) and a calibrated state (top right). On the bottom right only the reset potential was calibrated and an uncalibrated threshold taken. The reset was calibrated with a target of 0.2 V, while the threshold should be 0.6 V, resulting in an expected difference of 0.4 V. For the uncalibrated state the mean of the calibrated values was taken to get a comparable voltage mean and jittered with 4 bit to prevent crosstalk problems.

After calibration, the distribution is much more narrow as one can see on the top right. The standard deviation of the values before calibration was 11.7 %, while after calibration these numbers were brought down to 1.9 %. A standard deviation of 9.0 % was found when only calibrating the reset potential. The mean for both potentials calibrated is 0.390 V and so the result differs 2.6 % from the expected result.

The possibility to calibrate towards a wide range of target values is further shown on the bottom left in figure 3.4. Different target values for the threshold were given, the reset was always calibrated to be 0.2 V. Afterwards, the calibration result was determined and the mean and standard deviation plotted. In the visualization the standard deviation is smaller than the size of the points, so all neurons have a similar reset and threshold. For all data points the target was underestimated by about 5 %.

3.4. Membrane time constant

Calibrating time constants on the PPU is much more difficult compared to voltages. They are typically estimated from digitized traces by means of fitting an exponentially decaying function. This is difficult to perform on the PPUs due to the low

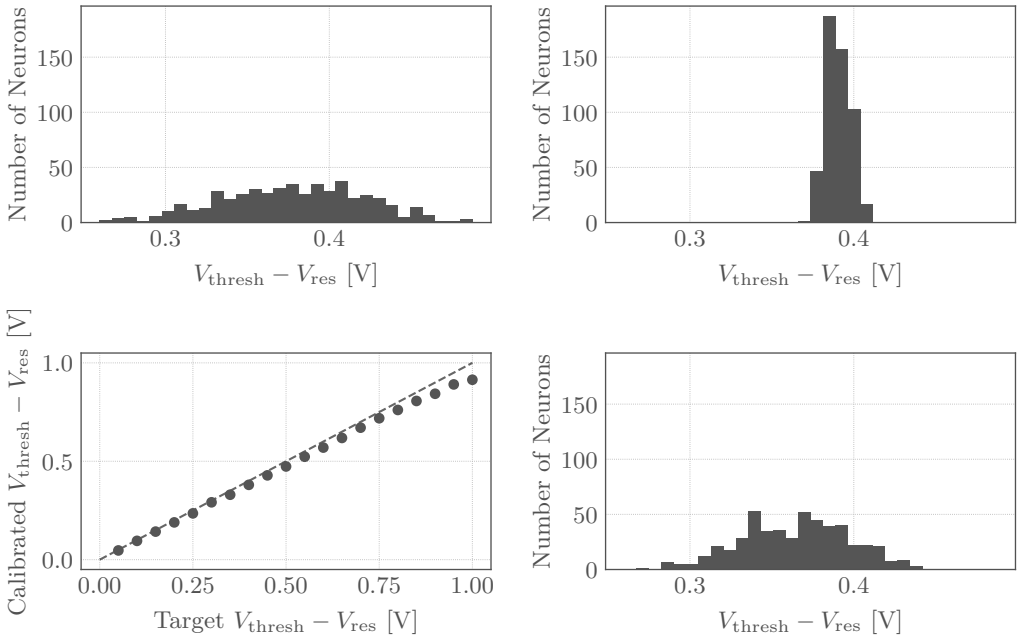

Figure 3.4: Histograms showing $V_{\text{thresh}} - V_{\text{res}}$ before (top left) and after (top right) calibration for all 512 neurons of both parameters. On the bottom right the difference for a calibrated reset and an uncalibrated threshold is visible. The desired target value was 0.4 V, with $V_{\text{res}} = 0.2\,\text{V}$ and $V_{\text{thresh}} = 0.6\,\text{V}$. The distribution of the voltage difference is narrowed down by calibration and is close to the target value. For the uncalibrated CapMem values the mean of the calibrated ones are taken and jittered to prevent crosstalk problems, to ensure a comparable voltage mean. On the bottom left a sweep of different target values and the resulting calibrated value is plotted with mean and standard deviation. The dashed line marks the desired target results. Data taken from chip 22 on setup 63.

sampling rate of the CADCs and the limited computational power and resolution of the PPUs vector units.

To calibrate the membrane time constant, the bias of the leak OTA, acting like a resistor connecting the membrane to the leak potential has to be calibrated. The time constant is the ratio of the membrane capacitance and the leak transconductance value ($\tau_{\text{mem}} = C_{\text{mem}}/g_{\text{leak}}$). This membrane capacitance is coarsely configurable, so changing this parameter makes a new calibration necessary. The membrane time constant is calibrated by changing the leak OTA bias.

For longer time constants a single timed CADC read can be used for estimation. After a forced neuron reset, the membrane potential will decay exponentially from the reset voltage back to the leak potential. After a time of τ_{mem} the amplitude has reached $1/e$ of the original amplitude, so the voltage will be

$$V_{\text{tau}} = V_{\text{leak}} - (V_{\text{leak}} - V_{\text{res}})/e. \tag{3.1}$$

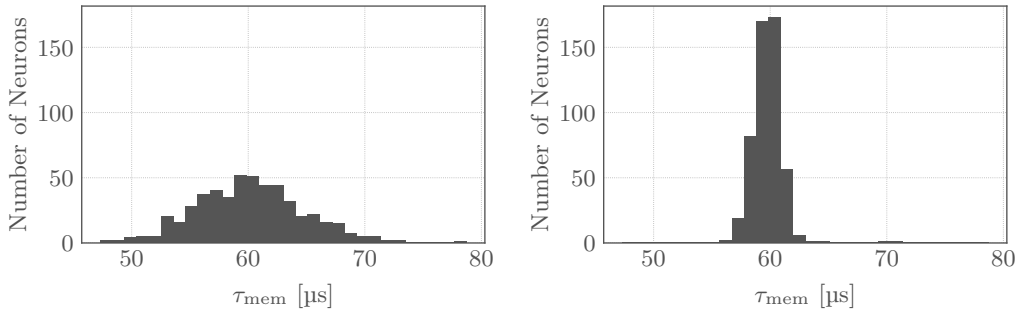

Figure 3.5: Histograms showing the membrane time constant of all 512 neurons before (left) and after (right) calibration for a target value of 60 µs. The distribution of the calibrated state is around the desired target and much more narrow compared to the uncalibrated distribution. To get a comparable mean the mean of the calibrated values was taken and jittered for the measurement of the uncalibrated values. Chip 22 on setup 63 was used for measurement.

With a known V_{res} and V_{leak} this expected value can be calculated. By resetting the neuron and triggering the CADC right after a wait of $\tau_{\mathrm{refr}} + \tau_{\mathrm{mem}}$ should then yield V_{tau}. If the voltage is higher, the membrane time constant is too short as the leak is reached faster. In case the voltage is lower, the membrane time constant is too long, yielding an observable for τ_{mem}.

To obtain membrane time constants which are adjustable over two orders of magnitude, each leak OTA has an individual current multiplication and division mode. Each scales τ_{mem} approximately by an order of magnitude. While with the division mode orders of 100 µs are reached, the multiplication mode enables time constants around 1 µs. The behaviour of the leak OTA was already investigated by Dauer [2020]. For membrane time constants above approximately 20 µs the division mode has to be used, while for times below 2 µs it is the multiplication mode.

For calibrating longer membrane time constants one has to differ between division and normal mode. To ensure that a calibration routine only takes the desired time constant and calibrates to a desired value, also the division bit has to be calibrated. This has to be done before the CapMem calibration using a noisy binary search. So first of all the CapMem cells are set to (50 ± 8) LSB in normal mode, where we expect time constants of approximately 20 µs. With the according wait time for the desired τ_{mem} the CADC is read out. If the observed value is above the target, it shows that the longest possible time constant in normal mode is still to fast, thus the division mode is required to get longer membrane time constants. Otherwise the normal mode is used.

The difference between calibrated and uncalibrated membrane time constants is shown in figure 3.5. The target in this case was set to 60 µs and therefore all neurons are in the division mode. To measure the uncalibrated state, the mean of the calibrated values was taken and jittered with 4 bit. This was done to get a comparable mean, which is also visible in figure 3.5. To measure the actual time constant, the neuron was forced to the reset value and the exponential decay back to the leak potential was observed with the MADC. By fitting an exponential function

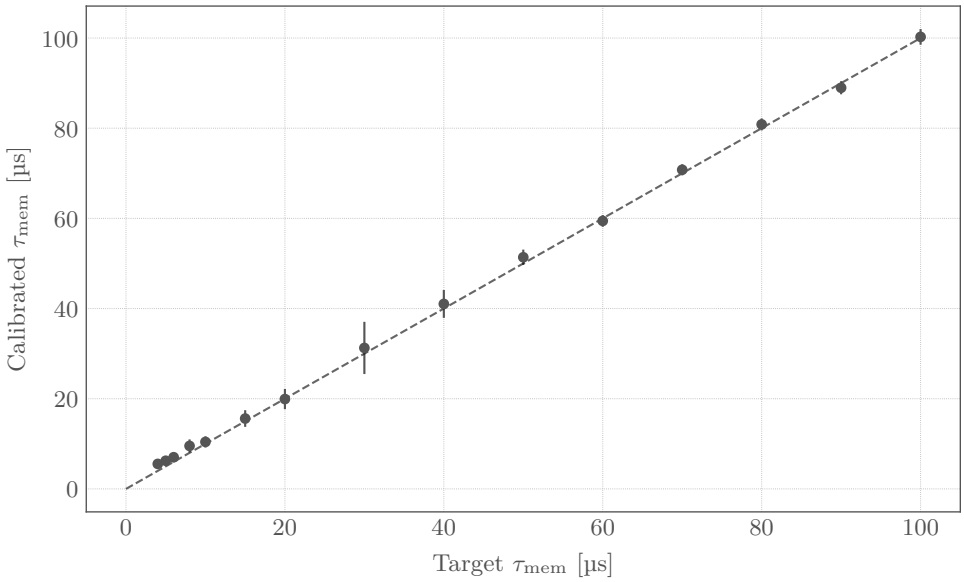

Figure 3.6: Sweep of different target values and the resulting calibrated membrane time constant for values between 5 µs up to 100 µs. For calibration we used a timed CADC conversion after τ_{mem}. The mean and standard deviation for all 512 neurons are plotted as black data points. The dashed line indicates unity, i.e. the targeted values themselves. Data taken from setup 63 on chip 22.

to the obtained trace data τ_{mem} was extracted. This MADC-based method serves as verification for the PPU based calibration, because it is a more precise estimate.

With calibration (right) the distribution is much more narrow compared to the histogram with the uncalibrated data (left). The mean of the calibrated values is 59.75 µs which matches perfectly with the desired target value of 60 µs. Without calibration the standard deviation is about 7.6 %, brought down to 2.1 % after calibration.

We furthermore attempted to quantify the quality of our verification method. For this purpose, we repeated the same MADC-based measurement 50 times. Across these trials we observed a standard deviation of 1 %. These deviations can result from noise on the membrane potential and the resulting differences in the fit.

To ensure that the calibration works for a wide range of membrane time constants a sweep was done for different target values, which can be seen in figure 3.6. Different target values from 5 µs up to 100 µs were used for testing and the mean and standard deviation of all neurons plotted. All values match the target value, showing that it is possible to calibrate to a desired τ_{mem} given in SI-units on the PPU. However, for values around 30 µs we observed increased deviations from the target. In this region one switches from normal mode to division mode for each neuron individually. Neurons with a time constant shorter than expected with settings of 50 LSB are put into division mode to get higher time constants. For some of these, however, even a setting of 1000 LSB in division mode yields a time constant about the target. This issue can be mitigated by improving the switching criteria and lowering the threshold

of 50 LSB. In this case, the characteristics of each CapMem quadrant have to be considered, as some instances exhibit unreliabilities for lower values [Dauer, 2020].

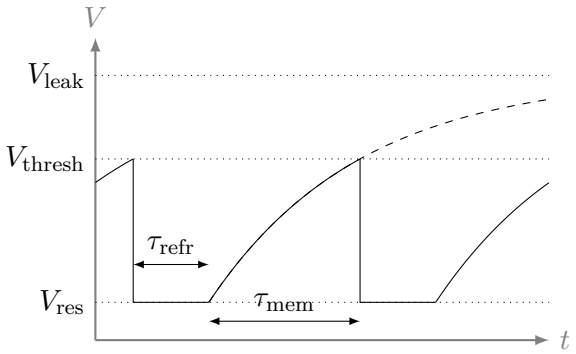

Figure 3.7: Sketch of a neuron with $V_{\text{thresh}} = V_{\text{tau}}$ (equation 3.1), resulting in regular spiking. After a spike occurred the neuron is in refractory mode. Then the membrane starts rising again until it reached V_{thresh} within τ_{mem}.

To calibrate even smaller time constants in the order of µs, we can not rely on above's method, since the CADCs sampling period itself fall in that range. Instead we can now rely on a leak-over-threshold configuration. By setting $V_{\text{thresh}} = V_{\text{tau}}$ (see equation 3.1), the neuron will spike regularly as it is sketched in figure 3.7. This allows to to calibrate τ_{mem} with the neuron spike counters. If the membrane time constant is perfectly calibrated the expected amount of spikes in an interval of $x \cdot (\tau_{\text{mem}} + \tau_{\text{refr}})$ should be x. For higher membrane time constants a lower amount of registered spikes is expected and vice versa.

For shorter time constants it is important to note that one has to differ between multiplication and normal mode. For time constants of approximately 2 µs and below the multiplication mode has to be used. So the multiplication bit has to be determined before starting with the calibration of the CapMem values. The same procedure as for longer time constants is used, all neurons are brought to 50 LSB (with noise) and multiplication is enabled. If the maximum reachable time constant is below the desired one, the multiplication bit is discarded.

To get an overview whether the method is also applicable for different time constants a sweep from 0.5 µs up to 10 µs was performed. As one has 8-bit spike counters it was decided to target for 200 spikes in an according interval. The results can be seen in figure 3.8. Most data points match the desired value within their standard deviation. In comparison to figure 3.6 the standard deviations are much higher. One explanation could be that this calibration method depends on three precalibrated values instead of two, so the total error increases due to error propagation. Especially for time constants around 3 µs it was observed that for some neurons the desired time constant was not be reached in either of the two modes, at least with the mode selection criterion specified above. In multiplication mode the longest time constant reachable was below 3 µs, but in normal mode time constants shorter than 4 µs were also not possible within the applied range. So the high standard deviation in this area is partly due to this problem. Like described above the characteristics of each quadrant have to be considered individual as done by Dauer [2020].

The offset voltage of the leak OTA unfortunately exhibits a slight dependency on

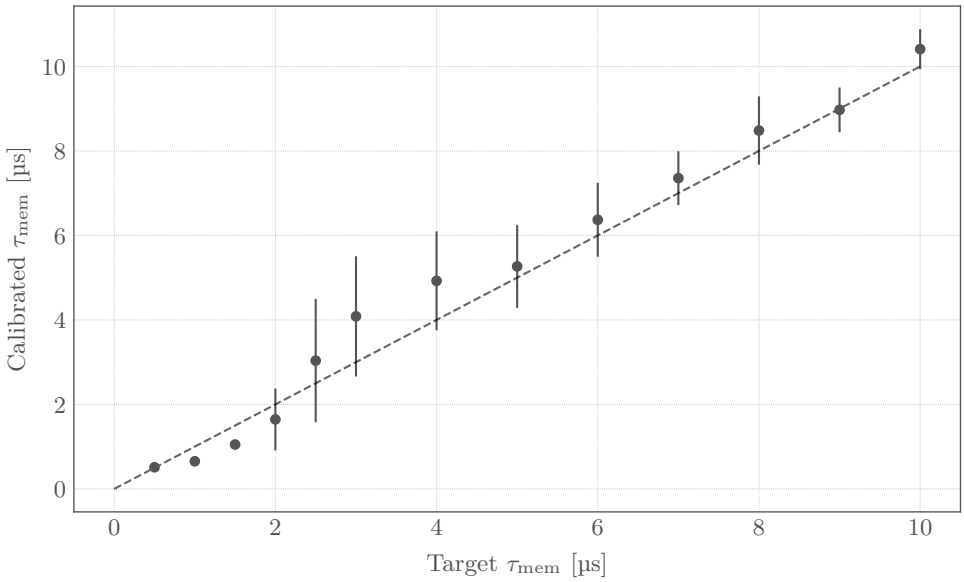

Figure 3.8: By using calibration with spikerates different target values from 0.5 µs up to 10 µs are used and the according calibrated τ_{mem} is plotted. Every data point is the mean and standard deviation for all 512 neurons on chip 22 (setup 63). The dashed line describes the expected values which should occur if the calibration is perfect.

the bias current, enough to distort the calibration of τ_{mem}. For this reason, while performing the binary search for both calibration methods, after every leak bias iteration the leak voltage has to be recalibrated. So in principle it is a nested binary search, in every iteration step another binary search is performed to calibrate V_{leak}. Thus the total amount of iterations is increased to 196 iterations in total compared to the 14 iterations for a noisy binary search for the whole parameter space.

3.5. Synaptic input

The heart of the synaptic input is an OTA, which generates a current onto the membrane depending on the difference between its two input potentials and its transconductance value g_{syn}. One terminal serves as reference voltage and should match the baseline of the other potential, which corresponds to of the synaptic input line idling at 1.2 V. Incoming spikes from the synapses now decrease the line potential, which results in a current output. The resistor-like element controlling the synaptic time constant pulls the line back to the idling potential. Every synaptic input has therefore three different parameters which have to be calibrated.

Depending on the excitatory or inhibitory nature of the instance, the charge Q onto or off the membrane, resulting from an incoming spike, depends on the amplitude on the synaptic input line as a function of the synaptic weight, the synaptic time constant and the gain of the synaptic OTA. The latter is determined by a bias current. The emitted current by the OTA depends only on the transconductance

g_{syn} and the voltage difference at its inputs:

$$I_{\text{syn, exc}}(t) = g_{\text{syn}} \cdot \left(V_{\text{ref, exc}} - V_{\text{syn, exc}}(t)\right). \tag{3.2}$$

An incoming spike at $t = 0$ with a drop of ΔV_{spk} will decay with a time constant of τ_{syn} back to the resting potential

$$V_{\text{syn, exc}}(t) = V_{\text{ref, exc}} - \Delta V_{\text{spk}} \cdot \exp\left(-\frac{t}{\tau_{\text{syn}}}\right), \tag{3.3}$$

resulting in a charge onto the membrane given by

$$Q = \int_0^\infty I_{\text{syn, exc}}(t)\,\mathrm{d}t = \Delta V_{\text{spk}} \cdot g_{\text{syn}} \cdot \int_0^\infty \exp\left(-\frac{t}{\tau_{\text{syn}}}\right)\mathrm{d}t = \Delta V_{\text{spk}} \cdot g_{\text{syn}} \cdot \tau_{\text{syn}}. \tag{3.4}$$

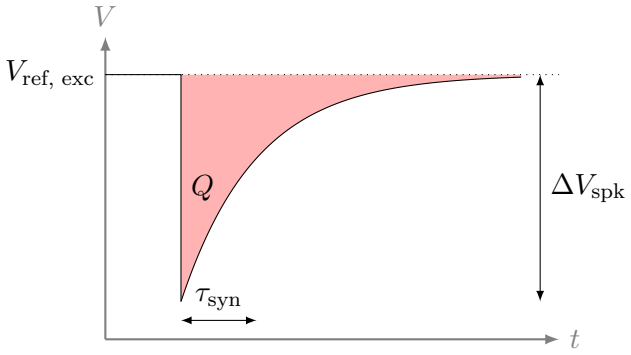

Figure 3.9: An incoming spike from the synapses drops the synaptic input line by ΔV_{spk}. It is decaying exponentially back to the baseline of 1.2 V with the synaptic time constant τ_{syn}. The total charge Q onto the membrane is now the integrated area in red.

Figure 3.9 shows a sketch of the synaptic input line with an incoming spike. The influence of the transconductance value is not visible as it is just responsible for translating the voltage difference to a current.

3.5.1. Reference voltage

With an optimally calibrated reference voltage, the synaptic input OTA should not produce any current when the input line is at its baseline of 1.2 V. With an uncalibrated synaptic input, a constant current is emitted. This changes the resting potential since the current needs to be compensated by the leak. For calibration this phenomena is used. First of all the membrane potential is read out with the CADC while the synaptic input is disabled. In this case the CADC value should be equal to the leak. By enabling the synaptic input, the membrane potential should change if the synaptic input emits a current. The goal of the calibration is now to reach the leak potential again with the synaptic input still enabled. This indicates that the OTA does not output any current.

This calibration works for the excitatory and the inhibitory input the same way, the only difference is the changed polarity. If the reference voltage is below the base-

line, the excitatory input OTA will emit a negative current onto the membrane while the inhibitory input OTA will emit a positive current and vice versa. This method already worked stable for different leak values on HICANN-DLSv3 [Leibfried, 2018] and also on the latest prototype it works fine.

3.5.2. Excitatory current

To calibrate the transconductance of the excitatory OTA, one could simply apply a fixed voltage difference at the OTAs inputs and – after letting the membrane converge to its equilibrium due to the leak – measure the resulting offset in the membrane potential. This would correspond to a calibration of the DC contribution. Instead we can absorb the capacitive mismatch in C_{mem} by calibrating g_{syn} over C_{mem}. For this purpose we configure a long membrane time constant and opt for a dynamic calibration using spikerates. Therefore, the synaptic input line can be pulled to a fixed voltage by an external DAC, resulting in a constant current onto the membrane according to equation 3.2. The voltage difference forced at the OTA inputs with the DAC is now called ΔV_{syn}. With $g_{\text{syn}} \cdot \Delta V_{\text{syn}} \cdot \Delta t$, the charge Q onto the membrane in a time interval of Δt can be calculated. Assuming one would neglect the leak, with the equation $\Delta V_{\text{mem}} = Q/C_{\text{mem}}$, describing the voltage change of the membrane potential when a charge Q is put onto it, one gets

$$\frac{C_{\text{mem}}}{g_{\text{syn}}} = \frac{\Delta V_{\text{syn}} \cdot \Delta t}{\Delta V_{\text{mem}}}. \tag{3.5}$$

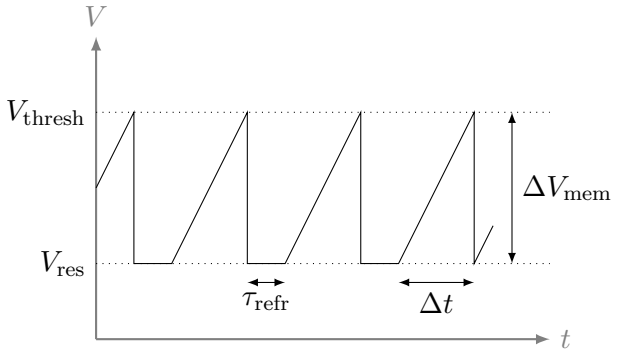

Figure 3.10: Sketch of a membrane potential stimulated with a constant current when the leak is neglected. The potential rises linearly until the threshold is reached, resetting the neuron to V_{res} for the refractory time. Afterwards the membrane potential again starts to rise.

While ΔV_{syn} can be easily adjusted by the external DAC one is left with ΔV_{mem} and Δt. Reading the raise of the membrane for a known time Δt however should not give reasonable results as the CADC will be too slow for readout. With a fixed ΔV_{mem} however it can be easily calibrated via spikerates. The principle is sketched in figure 3.10. Due to the voltage difference at the synaptic OTA a constant current controlled proportional to the gain is emitted onto the membrane. With a strong enough current the neuron will start to continuously emit spikes.

Calibrating to a certain amount of spikes in a certain time interval will now

calibrate the ratio of $C_{\text{mem}}/g_{\text{syn}}$. Because the spike counters can register 8-bit values, it was decided to use 200 spikes. So with a calibration 200 spikes should occur in a time interval of $200 \cdot (\tau_{\text{refr}} + \Delta t)$. Also the reset and threshold potentials have to be calibrated, which are set to 0.2 V and 0.6 V respectively, leaving a ΔV_{mem} of 0.4 V. The external DAC is set to 1 V, leaving a ΔV_{syn} of 0.2 V. So the desired ratio can be controlled by setting Δt and the quotient will just be $\Delta t/2$ according to equation 3.5.

In this entire consideration the leak was neglected. With a small membrane time constant the leak would have a big influence with this method. So beforehand τ_{mem} was calibrated to be 60 µs. With a Δt in the order of some µs it is justified to neglect the leak for this calibration. Of course the voltages can be adjusted if desired. With the synaptic input designed for maximum deviations of approximately 0.2 V, the chosen potentials make use of the whole dynamic range. The smaller the voltage at the synaptic input and the higher the difference between threshold and reset the influence of the leak gets higher and must be taken into account.

Before measuring the spikerates, the synaptic reference voltage has to be recalibrated. That is because the offset between the two inputs can depend on the bias settings. So a nested binary search is used and in every iteration step another binary search for calculating $V_{\text{ref, exc}}$ is performed. So a total of 196 iterations are performed. It is worth noting that also a CapMem wait is implemented before every calibration of the reference voltage to get a stable leak at the beginning.

3.5.3. Inhibitory current

Calibrating the inhibitory synaptic input gain with the same method as the excitatory is basically possible. For this instead of setting the external DAC pulling the synaptic input line to 1 V one could set it to 1.4 V, also resulting in a constant current onto the membrane. The synaptic input line however is just designed for voltages of 1.2 V and below, so this method is not used.

With the help of the excitatory input it still can be calibrated with another method. As already mentioned in this thesis, the membrane will idle at leak potential if no other inputs are enabled. By connecting both synaptic input lines to the same potential with an external DAC, the excitatory input will emit a current onto the membrane, while the inhibitory input pulls charge from the membrane. If the inhibitory input is stronger than the excitatory the membrane potential will be below the leak, while it will be higher for a stronger excitatory input. With both synaptic inputs enabled and a membrane potential equal to the leak, the gain of the two OTAs should be equal. The membrane potential before connecting the OTAs to the membrane can be read out with the CADC, marking the target. A binary search now finds a suitable transconductance value where the excitatory current is compensated by the inhibitory input. Compared to the excitatory calibration this is not a "dynamic measurement". It is better compared to the offset calibration by waiting for a steady-state and then measure the potential difference.

As in the excitatory calibration the reference voltage has to be recalibrated for every iteration step. To calibrate the inhibitory input with this method an already calibrated excitatory input is needed. If different gain values are desired, the excitatory gain firstly has to be calibrated to the desired inhibitory gain to calibrate the latter. Afterwards the excitatory gain can be re-calibrated to its desired value.

3.5.4. Excitatory synaptic time constant

The charge deposited onto the membrane by an incoming spike is given by equation 3.4. By neglecting the leak term, the synaptic time constant can be calculated with the resulting voltage raise on the membrane if the synaptic input is connected via

$$\tau_{\mathrm{syn}} = \frac{C_{\mathrm{mem}}}{g_{\mathrm{syn}}} \cdot \frac{\Delta V_{\mathrm{mem}}}{\Delta V_{\mathrm{spk}}}. \tag{3.6}$$

It was already shown on HICANN-DLSv3 [Leibfried, 2018] that it is possible to calibrate the synaptic time constant with spikerates. The same principle as with the excitatory gain for the OTA is used. By sending in spikes regularly and using a smallest possible refractory time, the amount of spikes in a certain time interval is proportional to τ_{syn}.

For the calibration two synapses with weight 63 are connected to each neuron and receive a poisson spike train with a frequency of 50 kHz from the background spike generators. The difference ΔV_{mem} of reset and threshold was kept from the calibration of the OTA gain with 0.4 V. In a time interval of 2.5 ms the amount of registered spikes is calibrated to a target value of 200 spikes.

As already shown by Weis [2020], the voltage drop ΔV_{spk} on the synaptic input line for events with maximum weight depends on the physical location on the chip. With the synapse drivers in the middle of each chip half, the neurons located closer to the drivers have higher voltage drops at the synaptic input compared to neurons far away at the border of the chip. So it is expected, according to equation 3.6, that the calibrated synaptic time constant for neurons closer to the drivers in the middle should be calibrated to a lower value. Figure 3.11 shows a measurement of τ_{syn} after calibration with the former described algorithm. It clearly shows the deviations depending on the location of each neuron. With only 32 neurons on DLSv3 this behavior was not visible. For verification, the synaptic time constant was obtained by fitting an exponential to the synaptic input after an incoming spike. This method is also error-prone due to fitting errors, but it is still a good measure of the calibration quality.

To counteract this difference a weight calibration was developed to ensure the same ΔV_{spk} for every neuron to get comparable synaptic time constants. For this purpose, the CADC can be used because it is also able to read out the synaptic input line. As the tunable-resistance R_{syn} influencing the synaptic time constant can not be completely switched off one has to set it to its maximum value. Otherwise with time constants in the range of some µs the CADC, taking approximately 1 µs for a readout, is too slow to reliably measure ΔV_{spk}. For the calibration, again two synapses are used per neuron, with one set to 63 and the other to a value of 32. The latter synapse is now tuned such that the difference of a CADC read before a spike and right afterwards is the same for all neurons. With the tuned synapse weights it should be now possible to counteract the differences in each quadrant for the time constants.

In figure 3.12 the calibrated τ_{syn} for each neuron depending on its location on the chip with calibrated synapse weights is plotted. Within each quadrant the neurons now have a comparable synaptic time constant, the standard deviation within each quadrant is around 3.3 %. Compared to the calibration with uncalibrated synaptic weights the time constants are calibrated to higher values. This is due to the lower

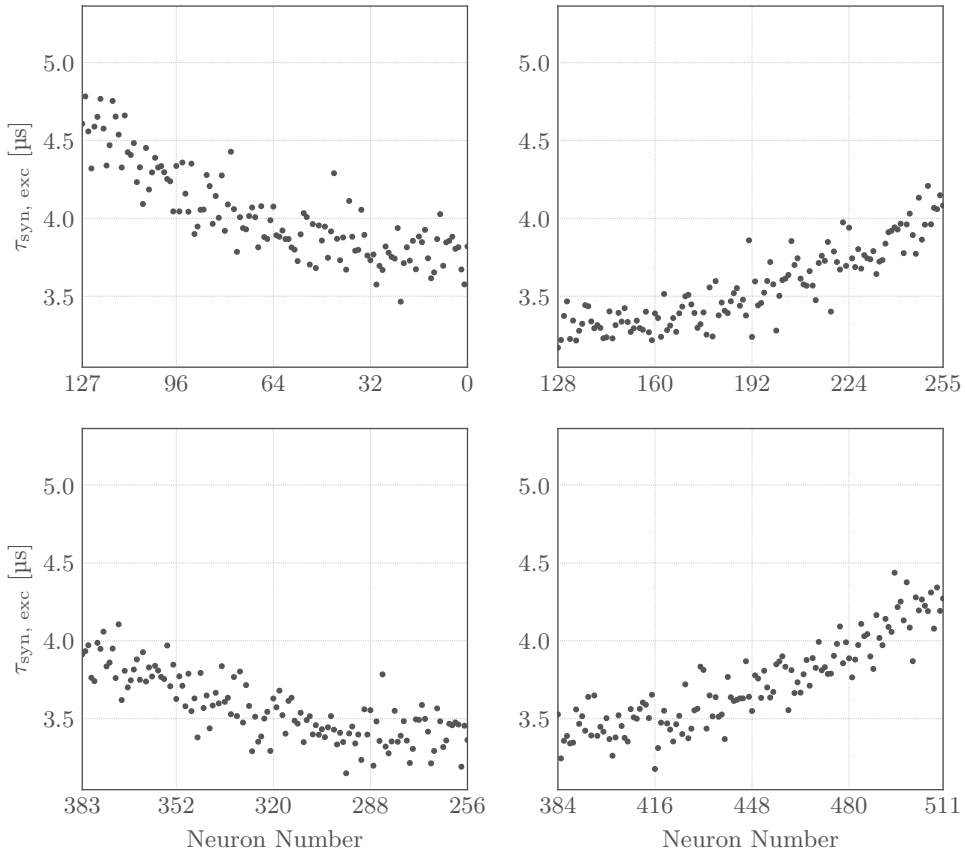

Figure 3.11: Calibrated τ_{syn} with spikes incoming from two synapses of maximum weight. The four subplots represent the hardware layout in four quadrants with the drivers located in the middle of each chip half (upper and lower). Center neurons are calibrated with a lower synaptic time constant because the drop at the synaptic input ΔV_{spk} is higher. Data was measured on chip 22, setup 63.

total weight resulting in a lower ΔV_{spk} and thus a higher time constant. Comparing the neurons across a chip shows large differences between the quadrants. In quadrant 0 (top left), for example, all calibrated time constants are in the order of 1 µs higher compared to the other quadrants. Aiming for same time constants chip-wide, and also on different chips, this accuracy is not sufficient.

One reason for the inaccuracies can be the slightly different resistance of the resistor-like elements. With a CapMem value set to zero the resistance should be at its maximum. Due to different CapMem ramps in each quadrant this value is expected to slightly differ [Dauer, 2020]. With a spike charging the synaptic input line not simultaneously but with a time of 4 ns a lower resistor would already allow a lower ΔV_{spk} because more charge is already pulled from the line. This should however be just a small effect with time constants expected to be on the order of 10 µs. A bigger influence could have the sampling time of the CADC, taking around 1 µs. With smaller resistances the synaptic input is pulled back faster to the baseline, thus the "real" ΔV_{spk} is calibrated higher.

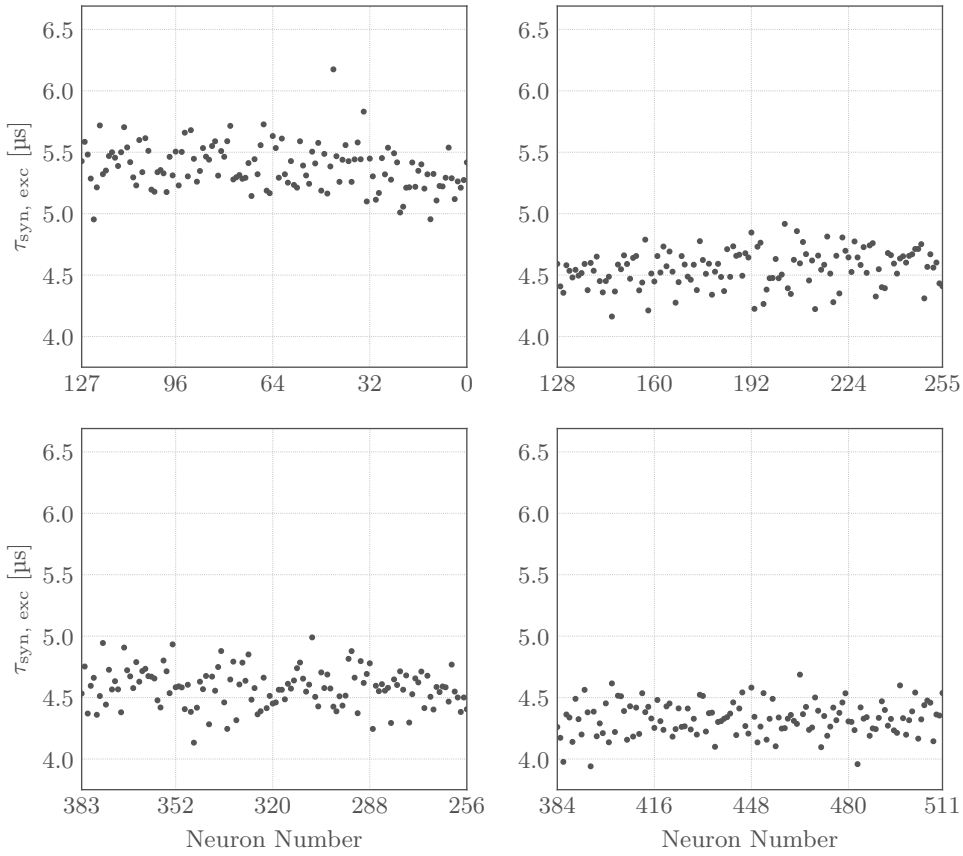

Figure 3.12: τ_{syn} is calibrated via spikerates with former calibrated synaptic weights to ensure the same ΔV_{spk} for all neurons. The four subplots show the hardware layout forming four quadrants. In each quadrant all neurons have a comparable synaptic time constant, while by comparing different quadrants with each other this is not the case. Data taken from chip 22, setup 63.

While calibrating the synaptic time constant with spikerates works in principle, it still has some drawbacks. With the presented method its working for moderate time constants around 5 µs. For longer time constants saturation could occur on the synaptic input. This can be however circumvented by using different input rates depending on the desired time constant. But there is still the problem that the refractory time is set to 1.28 µs. During this time signals on the synaptic input line do not affect the membrane. For the calibration of the OTA's gain this does not pose a problem because the synaptic input line voltage is fixed. With a dynamic line potential information is lost, which can be a bigger problem for smaller τ_{syn}.

Besides these issues it also lacks the possibility to calibrate the synaptic time constant to specified targets, the problem is the unknown conversion factor of spikerates corresponding to SI-values. That is the case because ΔV_{spk} cannot be determined properly with the CADCs. The routine still can be used to counteract the mismatch between different neurons in a quadrant to calibrate neurons to match an already calibrated one.

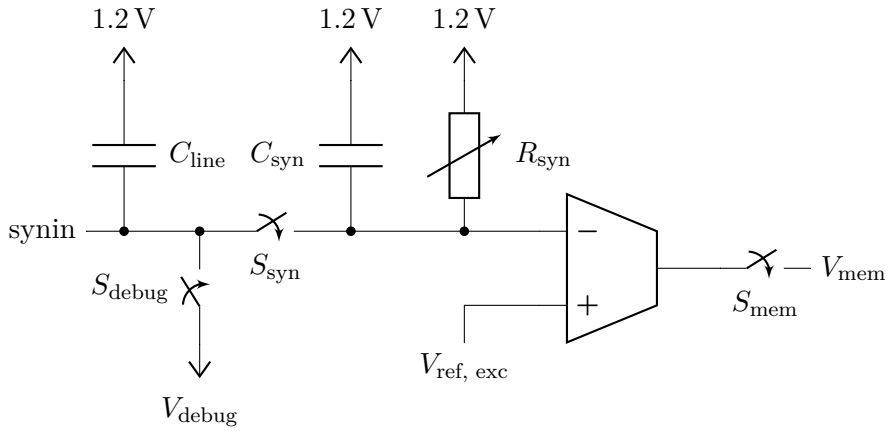

Figure 3.13: Simplified schematic of the excitatory synaptic input (looks the same for the inhibitory input with changed OTA polarity). Spikes coming from the synapse array are inserted on the left. These are charging the parasitic capacitance C_{line} and, in case the switch S_{syn} is closed, also the small capacitor C_{syn}. These are charging the parasitic capacitance C_{line} and an additional small capacitance C_{syn}. The switch S_{syn} lies in between. Via the tunable resistance R_{syn} the line potential is pulled back to the supply voltage of 1.2 V. A current proportional to the voltage difference of the reference voltage and the line potential is then emitted to the membrane via an OTA through the switch S_{mem}. Via the debug pin, separated by the switch S_{debug} from the input line, an external voltage can be connected.

So a different approach was developed to force a consistent ΔV_{spk} for every neuron independent of the quadrant or even chip. Figure 3.13 shows a simplified schematic of the synaptic input and surrounding circuits.

With a closed S_{syn} the synaptic input could now be charged to a desired ΔV_{spk} with an external DAC via the debug input. By previously opening S_{mem} the membrane would be unaffected by the resulting current. By opening S_{debug} and closing S_{mem} simultaneously, the membrane capacitor could be used to integrate the decaying signal from the synaptic input by reading the membrane potential with the CADC. In contrast to the other switches, S_{mem} can not be switched via the vector unit resulting in an inefficient scaling and imprecise timing, since the switching has to be done individually for every neuron

But via another method the external DAC can still be used to generate a fixed ΔV_{spk}. By closing S_{debug} and opening S_{syn} the line capacitance C_{line} can be precharged to a voltage provided by an external DAC. After opening S_{debug} , closing S_{syn} will result in charge being shared between the precharged C_{line} and C_{syn} which was pulled to 1.2V by R_{syn}. The voltage will result as

$$\Delta V_{\text{spk}} = \left(1.2\,\text{V} - V_{\text{debug}}\right) \cdot \frac{C_{\text{line}}}{C_{\text{line}} + C_{\text{syn}}}. \tag{3.7}$$

Considering extracted parasitics, we estimate $C_{\text{line}} = 590\,\text{fF}$ and $C_{\text{syn}} = 481\,\text{fF}$. With this method “fake” spikes can be generated with amplitudes independent of

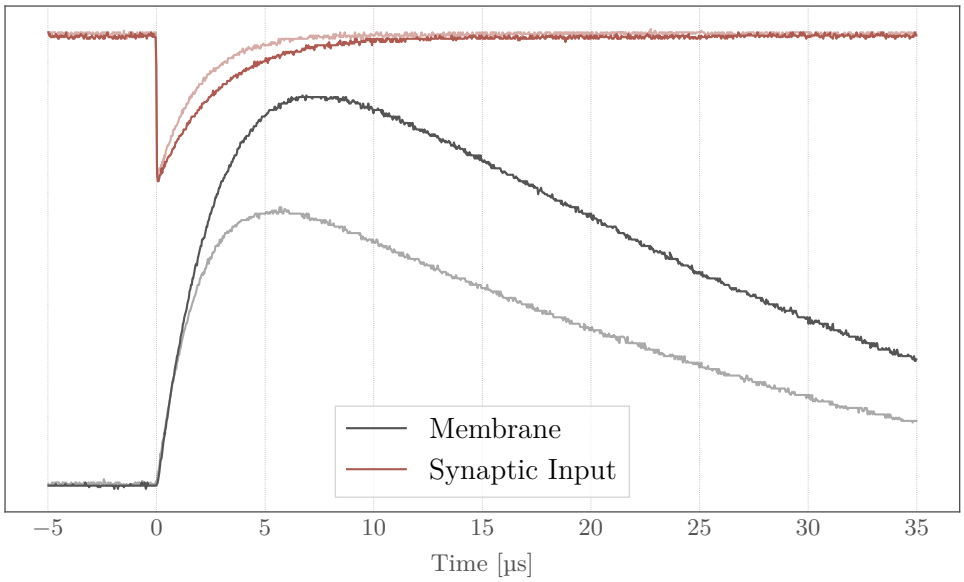

Figure 3.14: Traces of the membrane and synaptic input. The signal on the synaptic input is generated via a fake-spike at $t = 0$ and decaying back to its baseline afterwards, while the OTA emits a current onto the membrane resulting in a rise of the membrane potential. Due to the leak the membrane potential slowly decays back to the leak voltage. The strong black and red curve belongs to a R_{syn} bias value of 100 LSB resulting in a longer synaptic time constant compared to the light black and red curve belonging to a bias value of 150 LSB and a shorter time constant. With longer time constants the membrane potential is higher. Neuron 0 was measured on setup 63 with chip 22.

instance, quadrant and even chip. The integration of a fake-spike on the membrane is shown in figure 3.14. Starting from a low leak potential, the membrane starts rising due to the signal on the synaptic input. It is clearly visible that for a longer τ_{syn} the rise in the membrane potential is higher compared to shorter time constants. But also the leak has an effect pulling the membrane potential back. Considering the synaptic input's parameters as well as the membrane dynamics we find

$$\Delta V_{\text{mem}} = \frac{\Delta V_{\text{spk}} \cdot \tau_{\text{syn}}}{\frac{C_{\text{mem}}}{g_{\text{syn}}} \cdot \left(1 - \frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)} \cdot \left(\exp\left(-\frac{\tau_{\text{syn}} \cdot \ln\left(\frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)}{\tau_{\text{syn}} - \tau_{\text{mem}}}\right) - \exp\left(-\frac{\tau_{\text{mem}} \cdot \ln\left(\frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)}{\tau_{\text{syn}} - \tau_{\text{mem}}}\right)\right). \tag{3.8}$$

The derivation of this equation can be found in appendix A.4. By neglecting the leak, i.e. $\tau_{\text{mem}} \to \infty$, one arrives at equation 3.6. By continuously reading the membrane potential with the CADCs and only taking the maximum read should then deliver a suitable estimate of ΔV_{mem} even with a sampling period of 1 µs. Of course the leak potential has to be deducted from the maximum value. This procedure was

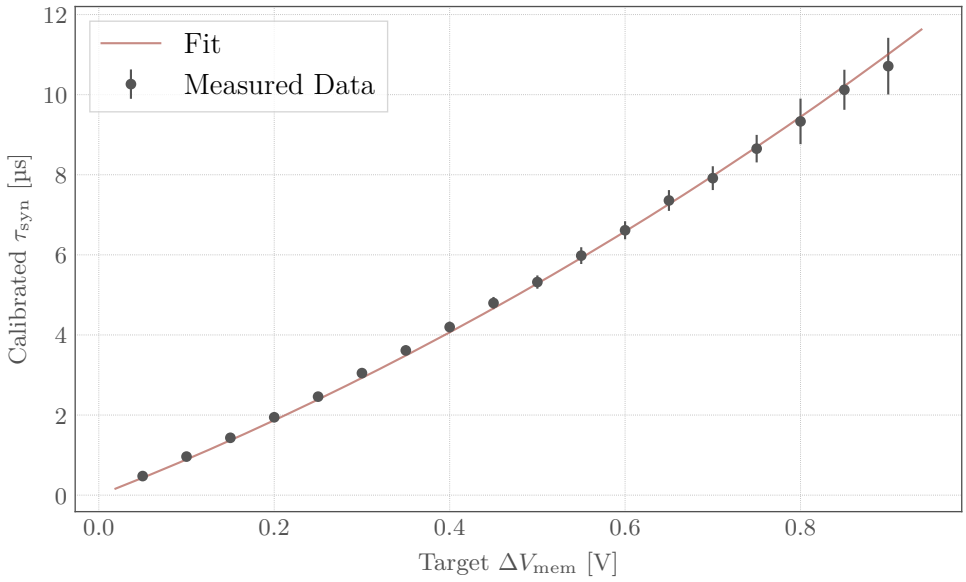

Figure 3.15: τ_{syn} was calibrated using different target ΔV_{mem}. The mean and standard deviation of all 512 neurons on chip 22 (setup 63) are plotted as data points. In red the curve of a fit according to equation 3.8 is visible. The fitted curve describes the course of the data points very well and all points match within the standard deviation.

completely implemented on the vector unit following the same algorithm already used for the threshold calibration (further described in appendix A.3). Again, a noisy binary search was used to calibrate τ_{syn}.

The exact correspondence between τ_{syn} and ΔV_{mem} was determined by sweeping the latter and measuring the former by means of the MADC and fitting an exponential decay. In particular, we took the mean across all neurons. The results can be seen in figure 3.15. Furthermore, equation 3.8 was fitted to the given data also included in the plot. For the fit of equation 3.8, we inserted all known or previously calibrated values, resulting in ΔV_{spk} as the last free parameter. The ratio $C_{\text{mem}}/g_{\text{syn}}$ was previously calibrated to 1 µs and the membrane time constant to 60 µs. The leak was calibrated to a low voltage of 0.2 V to ensure a high dynamic integration range without the occurrence of saturation. Fake-spikes were generated with an external voltage of 1 V.

According to the fit, a ΔV_{spk} of 0.12 V occurs in this scenario. According to equation 3.7, the ratio $C_{\text{line}}/\left(C_{\text{line}} + C_{\text{syn}}\right)$ can be calculated for the fit to 0.60. From extracted parasitics this value can be calculated to be 0.55. Both values slightly differ, which can be explained with various reasons. Firstly, the assumption was made that the other parameters given in equation 3.8 are perfectly set without considering any errors. Also the extracted parasitics can have slight deviations compared to the real chip.

Especially in the range between 1 µs to 10 µs we observed a good correspondence between measured data and the fit function, which can now be used to determine a calibration target ΔV_{mem} according to a targeted time constant.

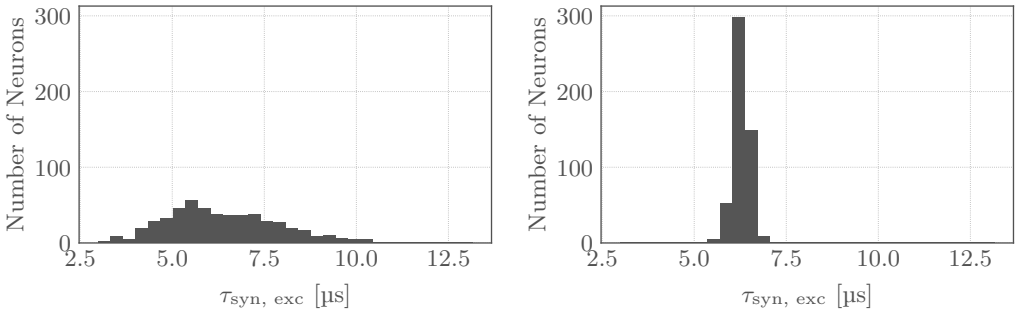

Figure 3.16: Histograms showing the excitatory synaptic time constant of all 512 neurons before (left) and after (right) calibration for a target value of 6 µs on chip 22 (setup 63). To get a comparable distribution with the same mean, the mean of the calibrated CapMem values was taken and jittered for the measurement of the uncalibrated values. The calibrated distribution is much more narrow and centered around the target value.

Calibration results for a target of 6 µs can be seen in figure 3.16. We take the mean of the calibrated CapMem values with noise to estimate the uncalibrated data shown on the left side. The standard deviation is in this case 23.8 %, including a outlier at 12.86 µs. After calibration (right) the standard deviation was brought down to 3.4 % and also the outlier would be calibrated. This is the same standard deviation as with the spikerate based calibration within each quadrant. The mean of all calibrated neurons is 6.27 µs, which is 3.8 % off the desired target value. Similar deviations can be also found in the host based calibration [Weis, 2020].

To ensure that this calibration method also works on other chips we repeated the sweep shown in figure 3.15. The results can be seen in figure 3.17. It shows that for a range from 0.1 V up to 0.7 V the different data points match perfectly. This corresponds to time constants from approximately 1 µs up to 8 µs. So by determining the desired ΔV_{mem} with the once determined fit of equation 3.8 can calibrate the synaptic time constant of all setups.

Outside of this boundary the standard deviation of some setups is increased compared to other setups. For example setup 73 has a large standard deviation for voltage differences higher than 0.8 V, which corresponds to a potential of 1 V considering the leak of 0.2 V. That is the case because the CADCs saturate differently on different setups, especially with the host-based calibration [Weis, 2020]. The calibration of the CADCs presented in chapter 3.2 was developed after the data for the sweep was acquired and the CADCs were calibrated with the routine of Weis [2020]. So for some neurons the time constant was calibrated too high, because of the saturation it was not able to get a higher ΔV_{mem} and so the time constant was further increased in the binary search. Effects like this can be easily circumvented by using a different debug voltage to optimize ΔV_{spk} and thus getting another rise in the membrane potential matching the CADC characteristics. Therefore another sweeps with different fits have to be executed.

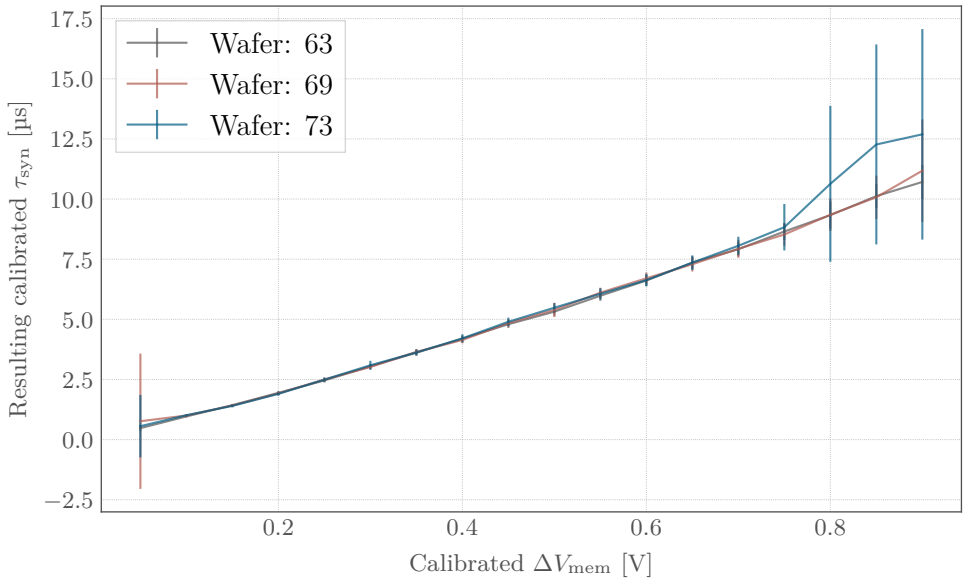

Figure 3.17: Calibrating τ_{syn} with different ΔV_{mem} as target on different chips/setups. Every data point marks mean and the standard deviation of all 512 neurons on the according chip. In black, the data was gathered from chip 22 on setup 63. Data from chip 30 on setup 69 is drawn in red, while chip 31 on setup 73 is plotted in blue. For target Δ_{mem} in the range of 0.1 V up to 0.7 V all curves match with each other. Outside of this boundary for some setups the standard deviation is clearly increased compared to other setups. For lower values this is especially the case for setup 69, while for higher values this is more the case of setup 73.

3.5.5. Inhibitory synaptic time constant

Calibrating the inhibitory synaptic time constant could work based on the same principle as the excitatory one. In principle one just has to determine the drop of the membrane potential instead of the raise. As already mentioned, the leak just can be set for some neurons to a maximum value of approximate 0.6 V. This would just allow a dynamic range of ΔV_{mem} up to 0.4 V, for lower voltages than 0.2 V the CADC will not be able to read it. So a different approach was used to calibrate the inhibitory τ_{syn} with the help of the excitatory synaptic time constant.

By sending in equally-sized fake-spikes to both synaptic inputs simultaneously, the membrane potential should rise if the excitatory synaptic time constant is longer than the inhibitory time constant (if the gain of both OTAs is identical). That is because the excitatory input puts more charge to the membrane. For $\tau_{\text{syn, inh}} > \tau_{\text{syn, exc}}$ however the membrane potential will drop. So like for the excitatory input the rise in the membrane potential can be determined with the CADCs. A value above zero indicates a stronger excitatory input, while a value around zero indicates a stronger inhibitory input. So a binary search has to find the point where the the value starts rising. Setting the target to a rise of 2 LSB gives a good guess for the inhibitory time constant, also including noise on the membrane. In this case the

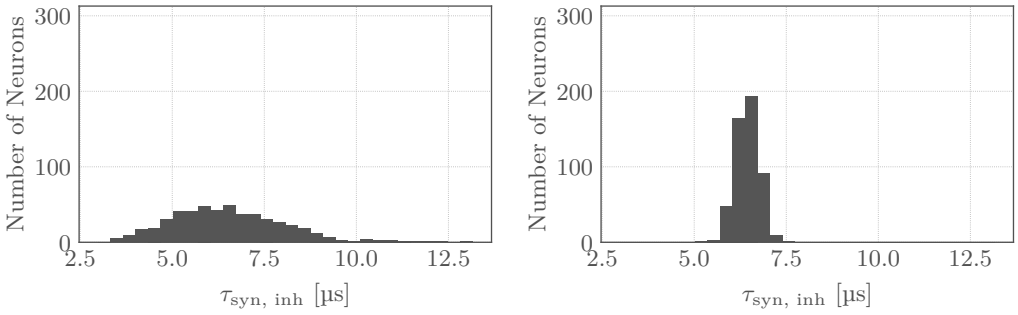

Figure 3.18: Inhibitory synaptic time constant of all neurons before (left) and after (right) calibration on chip 22 (setup 63). Calibrating the time constant narrows the distribution which is approximately centered around the target value of 6 µs. Uncalibrated CapMem values are determined by the mean of the calibrated values with noise to get a similar mean.

inhibitory input is barely able to counteract the excitatory input, thus showing both time constants are equal.

Figure 3.18 shows calibration results for a target value of 6 µs in histograms for all 512 neurons. Both fake spikes were generated with a debug potential of 1 V. Again the uncalibrated values are selected as a noisy mean of the calibrated CapMem values. In this case the standard deviation is about 24.0 %, which is equal to the measurement from the excitatory input, which makes sense as both circuitry are designed identically. Also a big outlier of 13.16 µs is contained in the dataset. Calibrating the inhibitory time constant with the method described above the standard deviation is brought down to 4.9 % with a mean of 6.43 µs.

Compared to the excitatory synaptic time constant the standard deviation is a little bit higher. This is expected as the whole calibration also depends on the results of the excitatory time constant. The standard deviation for $\tau_{\text{syn, exc}}$ was 3.4 %. Supposing that for a perfectly calibrated excitatory τ_{syn} the standard deviation for the inhibitory would be also 3.4 %, one can calculate by error propagation an error of 4.8 % taking into account that the excitatory time constant is also not perfect. This is exactly the standard deviation obtained by measuring. The systematic shift of the mean could be explained by a slightly different synaptic line capacitance C_{line} of the inhibitory input compared to the excitatory input. This could result from asymmetries in the layout resulting in different parasitic capacitance, hence influencing the amplitudes of the fake events.

The calibrated inhibitory synaptic time constant always depends on the value of the excitatory one. This is further shown by comparing both obtained means and standard deviations in a plot visible in figure 3.19. If the excitatory and inhibitory time constants match, they would lie on the dashed grey line. For the whole range we observed a good correspondence between calibrated excitatory synaptic time constant and the according inhibitory time constant. So we can conclude that it is possible to calibrate $\tau_{\text{syn, inh}}$ with the help of $\tau_{\text{syn, exc}}$.

Figure 3.20 shows different set target values for $\tau_{\text{syn, exc}}$ and $\tau_{\text{syn, inh}}$ and the resulting calibrated value. An external voltage of 1 V was used for this sweep to

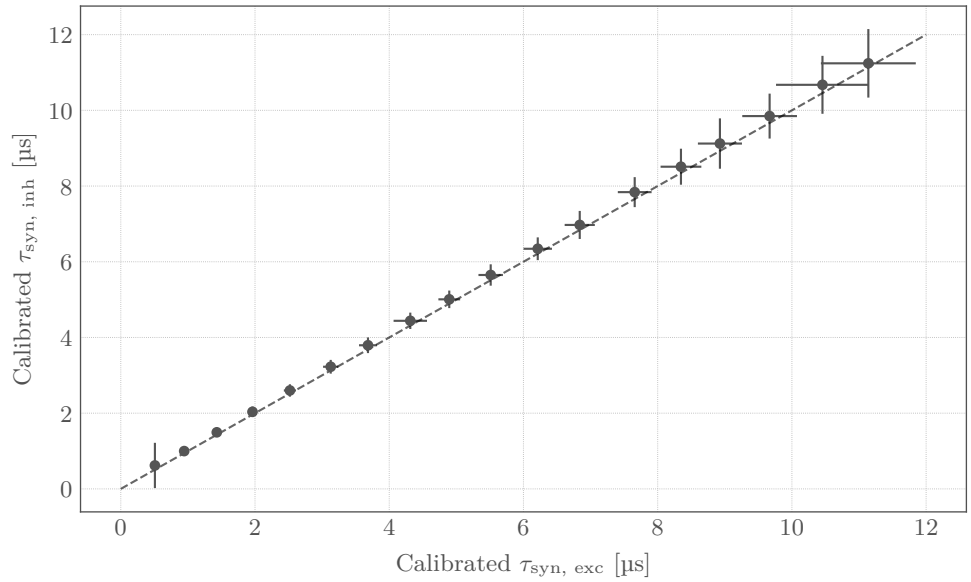

Figure 3.19: The excitatory synaptic time constant was calibrated and afterwards the inhibitory time constant was calibrated to the same target value. Every data point is the mean of $\tau_{\text{syn, exc}}$ and the according calibrated $\tau_{\text{syn, inh}}$ with the standard deviation of all 512 neurons on chip 22 (setup 63). The dashed grey line describes the point where $\tau_{\text{syn, exc}} = \tau_{\text{syn, inh}}$. The mean of the inhibitory time constants is always slightly above the grey line, so compared to its excitatory counterpart its always a little bit longer.

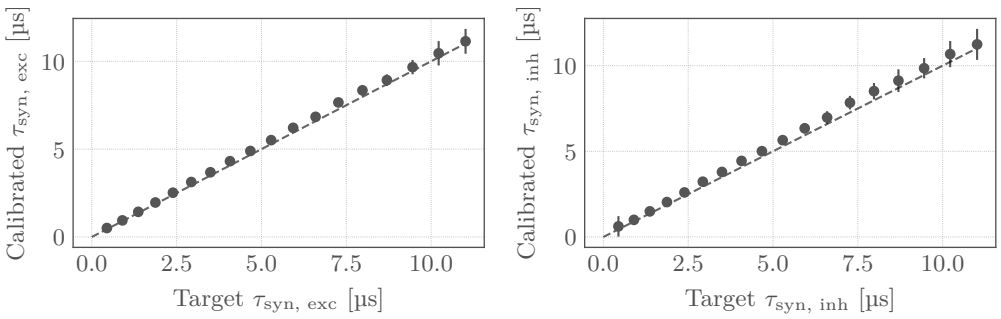

Figure 3.20: Sweep of different target values for the excitatory (left) and inhibitory (right) synaptic time constant and the resulting calibrated value. The mean and standard deviation of τ_{syn} for all 512 neurons on setup 63 on chip 22 is shown. Matching target and result is marked with the grey dashed line. An external voltage of 1 V was used to generate the fake-spikes used in the calibration. With this potential, time constants can be calibrated from about 1 µs to 10 µs.

generate ΔV_{spk}. For both calibrations low standard deviations occur, thus indicating with each step the neurons synaptic time constants are equalized. Also the calibrated

values match closely the target values, showing that it is possible to give SI-units and get the according value on hardware. With the used external voltage of 1 V an area from 1 µs up to 10 µs can be covered. As already mentioned, for shorter synaptic time constants a lower external potential should be used to get a better response on the membrane with a bigger ΔV_{spk}, while the standard deviation for longer time constants could be lowered by a higher external potential. Both would make a new fit necessary.

3.6. Performance and scalability

All calibration routines presented in this chapter are able to calibrate to a certain target value, including also the equalization of mismatch between different neurons. The runtime of individual routines never exceeded 5 s. In the following chapter the runtime of the individual calibration schemes are reviewed and also assessed according to their scalability for multi-chip systems such as wafer-scale platforms.

In order to ensure a reliable readout, we started by calibrating the CADC itself. The calibration is split into three parts taking 420 ms in total. The first and second part calibrate the CADC ramp and take approximately 204 ms each including 10 iterations. In each iteration a wait of 20 ms is implemented for the CapMem to reach a stable output. These waits dominate with a total contribution of 200 ms. Reading out the CADC and the according calculations are therefore done in about 4 ms. The last part calibrating the channel offsets just includes one CADC read and subtraction executed on the vector unit, so the runtime is only about 2.3 ms. Each of these parts require individual voltages supplied by the external DAC. However, programming and settling times do not pose a major contribution to the overall runtime.

The presented calibration routines form a rather complex dependency tree. This becomes especially apparent when considering the calibration of the inhibitory synaptic time constant, which is estimated by assuming all other parameters besides the threshold to be known (Still other parameters needed a calibrated threshold to be set). We hence analyzed the performance of this routine and all of its dependency. When targeting a set of LIF parameters different to the ones used for the $\tau_{\text{syn, inh}}$ calibration, the respective routines have to be called separately afterwards, adding to the total runtime of the calibration. The calibration sequence is visualized in figure 3.21. This flowchart includes the time for individual calibration routines, all together take about 16 s in total until $\tau_{\text{syn, inh}}$ is calibrated. To get the other parameters calibrated differently, one has to go the flowchart backwards accordingly.

First of all the reset potential must be calibrated. The reset is then used in the calibration for the membrane time constant (Calibration for long time constants with the CADC). This calibration also needs a calibrated leak, which is recalibrated after every iteration because the gain of the leak OTA influences the leak potential. Afterwards the leak is again calibrated but at the same voltage as the reset potential, which is important for the remaining calibrations. Afterwards the threshold is calibrated, together with the reset important for the calibration of the following excitatory synaptic input OTA bias. In every iteration the according reference potential is calibrated. With the calibrated bias also the synaptic time constant can be calibrated. Afterwards also the inhibitory synaptic input can be calibrated with the help of the excitatory one completing the LIF-neuron parameters.

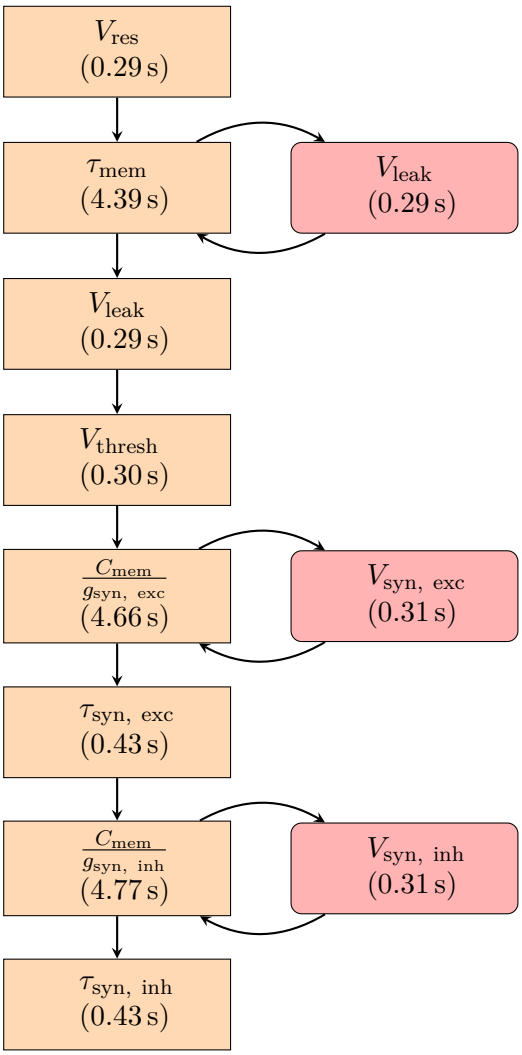

Figure 3.21: Flowchart showing the process of calibrating the inhibitory synaptic time constant and the other necessary LIF-neuron parameters to do so. The orange boxes are the main calibrations, while the red boxes with rounded borders are helper calibrations which are executed in every iteration of a main calibration. Every box also includes the runtime of each calibration. Starting with the reset the membrane time constant is calibrated together with the leak. By recalibrating the leak and with a calibrated threshold the bias of the excitatory synaptic input OTA can be calibrated together with its reference potential. Afterwards the excitatory synaptic time constant can be calibrated followed by the inhibitory synaptic input depending on all former calibrations.

Calibrating the potentials is quick as one can see in the flowchart, mostly taking about 0.3 s. The reset potential takes about 0.29 s performing a total of 14 iterations. Taking into account that after every iteration a wait of 20 ms is used for the CapMem to get in equilibrium, 0.28 s have already passed. So the rest, getting the observable and calculating how to set the bits, is done in only 0.01 s. A faster calibration just can be reached by revisiting the CapMem. The same applies to the leak potential also taking 0.29 s. A CADC read taking approximately 1.5 µs is four orders of magnitude smaller than the wait for the CapMem, explaining these differences. The calibration of the threshold potential takes a little longer with 0.30 s. Here the CADC is read out for 500 times being 3.7 % of one CapMem wait of 20 ms. This explains the slightly higher runtime. For the synaptic reference potentials the CADC is also read out just once, but the runtime is 0.31 s. The higher runtime can be explained that at the start of the calibration the leak potential as reference is read out with the CADC. Beforehand a wait of 20 ms is performed because the reference is calibrated within the OTA's bias calibration and a changed bias needs another wait until the true CapMem value is reached. So waiting for the CapMem takes 0.30 s in this calibration.

Calibrating the synaptic time constants takes a little bit longer with 0.43 s. With 14 iterations each, the wait for the CapMem is the biggest part with 0.28 s. The remaining time is mostly for getting the observable, calculating which bit must be set is just barely noticeable as this happens completely on the vector unit. Each observable resulted as the mean over 5 individual measurements. To ensure that the membrane potential is back to the leak after a former observation a generous wait is implemented after every fake spike. This ensures that even with a large time constant of 60 µs the leak is comfortably reached. Every observation, including the generation of the fake spike with CADC readout and the following wait, takes about 1 ms. Only half of the neurons can be covered with one vector unit instance. Determining the maximum in an array as explained in appendix A.3 takes 8 vector instructions. These extra vector instructions take about 0.3 µs each. With the CADC taking about 1.5 µs for readout this is a significant additional expense which is missing in the time critical measurement of the maximum CADC value. So the neurons are splitted, first the observable of the one half is determined followed by the other half. Summing up all single iterations, the time for getting all observables is about 10 ms. Getting the observables makes up the remaining time of 0.14 s. A neuron reset could make this generous wait redundant, pulling the membrane back the reset potential which should be in this case equal to the resting potential. Sending just one spike and determining the maximum read of all neurons at once would also be possible to reduce the runtime to be around 0.30 s, but at the expense of accuracy.

The calibration of the membrane time constant takes however 4.39 s. This is a nested binary search with a leak potential calibration in every iteration step. So 4.06 s are used for recalibrating the leak potentials. With 14 iterations also 20 ms for the CapMem ramp have to be taken into account taking another 0.28 s. The remaining 0.05 s are made up of calculating offsets and reading out the membrane potential with the CADC, taking approximately the time of the desired time constant. This could slightly change for different target values, but in comparison to the remaining runtime this would not be a significant change. In the flowchart also the

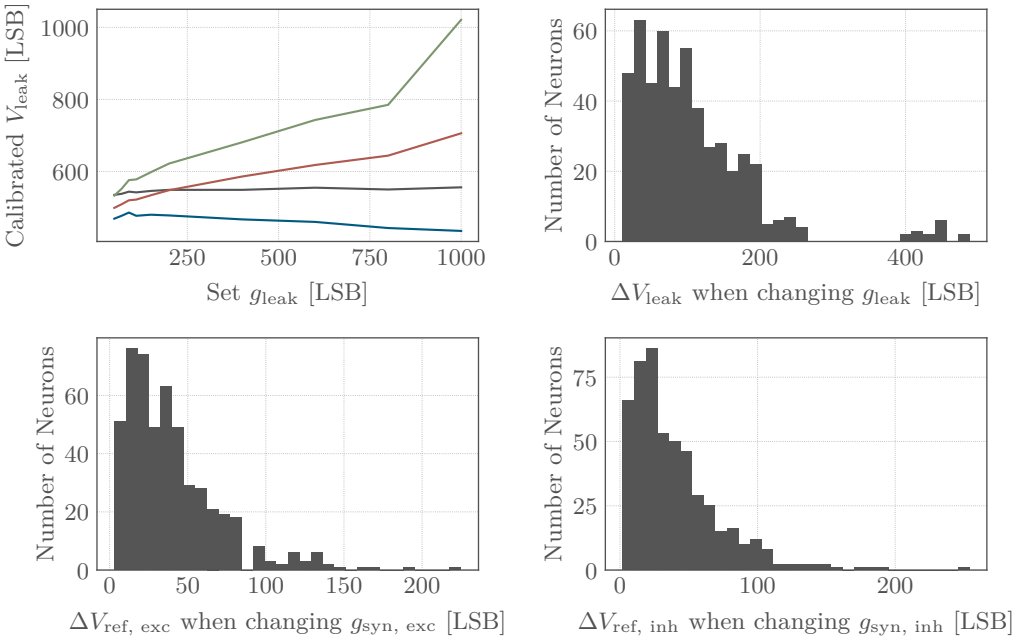

Figure 3.22: Dependency of the different OTA bias settings and the resulting calibrated input voltage in LSB. On the top left different leak bias settings are set (with noise) and the leak was calibrated. Four different neurons are shown with different dependencies. On the top right the difference of the maximum and minimum calibrated leak setting is put in an histogram for all 512 neurons. The same is done on the bottom. Here on the left the difference of the maximum and minimum calibrated reference potential of the excitatory input OTA is shown, while on the right the inhibitory input is visible. Data was taken on chip 22 on setup 63.

search for the division bit was not taken into account, because a long time constant was used for the further calibrations with division always enabled. Calibrating other desired membrane time constants the division bit search has to be done adding another 20 ms CapMem wait. For the calibration of short time constants via spikerates a similar runtime is expected. Due to a longer observation interval it should take a little bit longer, but not significantly longer.

The calibration of the OTA bias takes 4.66 s for the excitatory and 4.77 s for the inhibitory input. In both calibrations the reference potential is recalibrated in every iteration step. With 14 iterations each this are 4.34 s only used for the calibration of the reference potential. Additionally, the wait for the CapMem ramp has also taken into account with another 0.28 s. For the excitatory input only 0.04 s are therefore left used for getting the observables and calculation. The inhibitory bias calibration takes a little bit longer. Here beforehand another CapMem wait is performed because the baseline is determined with the CADC.

A big influence on the runtime has the recalibration of leak and reference potentials. So it was further investigated whether a complete recalibration is necessary. In figure 3.22 on the top left one can see the dependency of different leak OTA bias

settings and the resulting calibrated leak potential in LSB for four different neurons. For some neurons, the leak settings barely change as it is the case for the black curve. Other neurons need higher leak settings with an higher OTA bias, like it is the case for the red line or the other way around, like it is the case for the blue line. The green line shows a saturation effect. For some neurons with an bias of 1000 LSB the desired leak potential is due to the source follower and the saturated CapMem ramp not reachable. Thus the leak is set to its maximum value by the binary search.

Visualizing this for all neurons has been done in the histogram on the top right. The maximum difference of the calibrated leak potential values in LSB for sweeping the leak OTA settings from 50 LSB to 1000 LSB is visible. One can see that most neurons only differ by 100 LSB for the whole range. Some however have deviations above 400 LSB, which are all saturated like the green curve on the left explaining these big differences. In all of these cases the true leak potential is the same for 850 LSB and 1000 LSB, so it would not matter if the 1000 LSB are not reached. Dauer [2020] already showed that the leak potential is saturated for values of 850 LSB. The result shows that a 9-bit sweep, covering a range of 511 LSB, is enough to cover the whole range comfortably. So based on a calibration of V_{leak} at the start of the τ_{mem} calibration, a normal binary search with less iterations should be enough for recalibrating the former in every iteration of the latter. A noisy binary search would not be necessary as the already calibrated leak values still differ from each other, so instead of 14 iterations in every recalibration just 9 iterations are needed reducing the runtime significantly.

It is also not always necessary to execute a noisy binary search for the whole 10-bit τ_{mem} parameter space, as one could use the global mean of all chips as starting point and just do 8 iterations, which also covers all desired target values. Of course the 4-bit sweep at the end remains. The starting point of a desired target value still has to be determined beforehand with several sweeps. So 12 iterations compared to 14 iterations on cost of the flexibility could be done. Another positive aspect is that in this case the V_{leak} calibration also needs less sweeps because it does not have to cover the whole range of the leak bias OTA. With a reduced τ_{mem} range, a 7-bit sweep could be sufficient for recalibrating, so reducing the runtime by a half should be possible. In this case however, one also has to pay attention with the division/multiplication mode.

The two plots at the bottom show how the maximum and minimum value differ if the synaptic input bias current is sweeped for the whole range, with the excitatory on the left and the inhibitory on the right. The results are comparable to the leak OTA. So the same optimizations as with the leak OTA can be done here. With a reference potential calibrated at the start of the bias current calibration, for recalibration a 7-bit sweep should be enough for all neurons. So for all three OTA bias calibrations the runtime could be approximately halved by this method.

Another possibility is to reduce the 20 ms wait time for the CapMem ramp, because this time almost alone determines the entire runtime. Of course if the value is initially set from an unknown value before, these 20 ms are important to ensure a stable voltage or current. Otherwise the most significant bit could be set incorrectly making the remaining calibration unusable. But coming closer to the least significant bit, each CapMem cell reaches its equilibrium faster. In this case the 20 ms could be lowered. So using a wait time depending on the previous state can reduce

the runtime significantly. At every start of a calibration a wait time of 20 ms could be used, but with every bit further set the wait can be reduced by 1 ms. Also for the 4-bit binary search at the end of the noisy search just small waits around 10 ms can be used because the CapMem voltages or currents do not change that much with such small settings. Using this could reduce the runtime of all CapMem based calibrations. As the settings of the CapMem approximate the true value exponentially, this has to be further investigated if it makes routines more inaccurate. If that is the case, one has to discuss if the faster runtime excuses slight calibration deviations.

Using both optimizations as presented, the runtime for the whole LIF calibration could be brought down to less than 15 s for the whole chip. These 15 s include also the recalibration of the membrane time constant and the different potentials. Desired values for these parameters often vary from the value needed for the synaptic time constant calibration. So this recalibration always has to be considered.

This runtime will be also the same for multi-chip systems including many HICANN-X chips, as every HICANN-X calibrates itself as the amount of PPUs also scales with the amount of chips. Compared to the calibration of one wafer on BrainScaleS-1, taking 768 hours or 32 days [Schmidt, 2014], a runtime of 15 s for a potential BrainScaleS-2 wafer is a big improvement. The calibration is still very flexible and every neuron can be calibrated with individual parameters, because the target is given as an array. Such calibration speeds would also make calibration frameworks and databases obsolete, as they were used on BrainScaleS-1 [Müller et al., 2020], because for every experiment just a fast recalibration could be done.

3.7. Further calibration algorithms

The current calibration framework makes it possible to calibrate the whole biological parameters of the LIF neuron on the PPU as well as the CADC. Still other calibratable parameters exist on the chip, which were not presented in this thesis. On the biological side, also the adaptive exponential integrate and fire model (AdEx) extension to the LIF neuron need calibration. The possibility to do this was already investigated by Dauer [2020]. Also different multicompartment parameters have to be calibrated if one wants to use these features. The different potentials in this model can be read out with the CADC marking a potential observable for calibration algorithms.

It was also already shown that the synapse drivers can be calibrated with the PPU on DLSv3 [Leibfried, 2018] to use short term plasticity. Also calibrating the neuron more technically, that incoming spikes are integrated on the membrane, like it is done in the hagen mode [Weis, 2020], should be possible. Some experiments also need different technical calibrations. Getting a target output spikerate with a certain input rate, or depending on synaptic weights is an important calibration for the insects experiment in chapter 4. These calibrations also were done on the PPU and can be found experiment related in section 4.4. Other experiments could also benefit from such calibrations and it is shown that it is possible to do so on the PPU.

4. Insects: Accelerated emulation on hardware

The simulation of large areas of the brain is one of the main reasons for the development of neuromorphic hardware. As mentioned in section 2, the human brain has 100 billion neurons and its computational complexity far exceeds the computational capabilities of contemporary hardware. Insect brains however are much smaller, the brain of the honeybee for example contains about one million neurons with a volume of approximately $1\,\mathrm{mm}^3$ [Menzel and Giurfa, 2001]. Nevertheless, the cognitive abilities of insects are remarkable and manifold.

For example, it was shown that paper wasps of the species *Polistes fuscatus* are able to recognize the faces of other individuals [Chittka and Dyer, 2012]. This allows for building a social hierarchy through a series of one-on-one fights, afterwards individuals recognize their opponents avoiding the repetition of potentially costly battles. Other experiments showed that insects are even able to acquire complex behavior. Loukola et al. [2017] trained bumblebees to transport a small ball to a defined location to gain reward. This was not achieved by trial and error, but mostly by observing other bees which already acquired the skill. Even numerical competence is possible with the seemingly simple nervous system. Howard et al. [2019] showed that trained honeybees in a Y-maze are able to add or subtract one element of the sample stimulus. With yellow color they had to choose the way with one stimulus subtracted, while a blue color indicated an addition of one to the sample stimulus.

New measurement techniques even allow more insights to neural processes. For example it was made possible to map and reconstruct the whole synaptic connectivity of fruit flys of the species *Drosophila melanogaster* via electron microscopy [Zheng et al., 2018]. Based on such studies, different areas of insect brains can be further studied. One example is a neural model, which is able to reproduce the path of a bee returning to its nest after searching for food [Stone et al., 2017]. Such small models can now be implemented on neuromorphic hardware for further investigation.

This model was already implemented by Schreiber [2021] on the older chip version HICANN-DLSv2. The whole environment and the body was simulated on the PPU, while the neural model was implemented on the analog core. Sensor and motor complexes were implemented on the PPU connecting to the physical neuromorphic network. For this purpose we developed an experiment-specific calibration scheme and the network was implemented on HICANN-X using its new features like the background spike generators.

4.1. Path integration

In times past, sailors had to navigate without GPS. To avoid getting lost on the ocean without landscape features, a way had to be found to determine the position. Estimating the position relative to a departure point by keeping track of the distance covered and direction of travel is called dead reckoning [Rogne et al., 2016]. Inertial navigation systems in ships use this mechanism to determine their location relying on compass information paired with speed measurement or acceleration data. Deviations in these data is summed up over time, thus exact measurements and data interpretations are necessary.

It was shown that different insect species are able to navigate based on the same

principle, referred to as path integration [Heinze et al., 2018]. Different ways have been evolved in different species to gather compass information. Some species developed the "human way" of getting compass information, using the magnetic field of the earth [Riveros and Srygley, 2008]. But also the polarized skylight [Schwarz et al., 2011], the position of the sun [Wehner, 1984] or even the moon [Dacke et al., 2004] are used in other species gathering the required compass information.

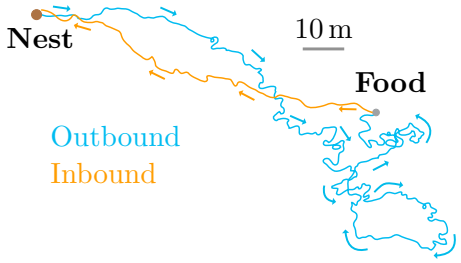

Figure 4.1: The foraging journey of the desert ant *Cataglyphis fortis* using path integration. Searching for food, it is travelling 354.50 m (blue path) starting from its nest. After food is found it directly returns home (orange path) travelling 113.20 m. Image taken and modified from Heinze et al. [2018].

An example for the foraging journey of insects is given in figure 4.1. The journey was observed from the ant *Cataglyphis fortis* living in the Saharan desert, an almost featureless environment. For path integration it uses its internal compass and the travelled distance is determined by a pedometer [Ronacher, 2020], so it counts its steps. This was impressively revealed by lengthening (attaching stilts) or shortening the legs [Wittlinger et al., 2006]. With longer legs the ant walked in the correct direction, but overestimated the distance by walking past its nest. Smaller stepsizes however made the ant to expect the nest entry too early on its home trajectory, but still walked in the right direction.

While land-bound species are able to use pedometry, flying insects are not able to gather step information. Their speed measurement is mostly derived from visual cues [Srinivasan et al., 2000]. The exact mechanism transporting the raw optical input information inside the brain is still a topic of ongoing research [Yakubowski et al., 2016]. In honeybees, however, neurons can be found that provide information about the optical flow [Stone et al., 2017].

4.2. Neural model

Path integration in the brain of insects is done in an area called central complex [Le Moël et al., 2019] located in the very center of the brain. This region is highly conserved among various insect species doing path integration, like in locusts [Heinze and Homberg, 2007] and butterflies [Heinze and Reppert, 2011]. Fossils suggest that this brain region already was contained in arthropods 520 million years ago [Ma et al., 2012], thus showing it evolved for many species. It is further composed of 4 neuropilar substructures, the protocerebral bridge, the fan-shaped body, the ellipsoid body and the paired noduli [Popov et al., 2003]. Figure 4.2 shows an 3D reconstruction of the central complex of a female *Megalopta genalis* bee.

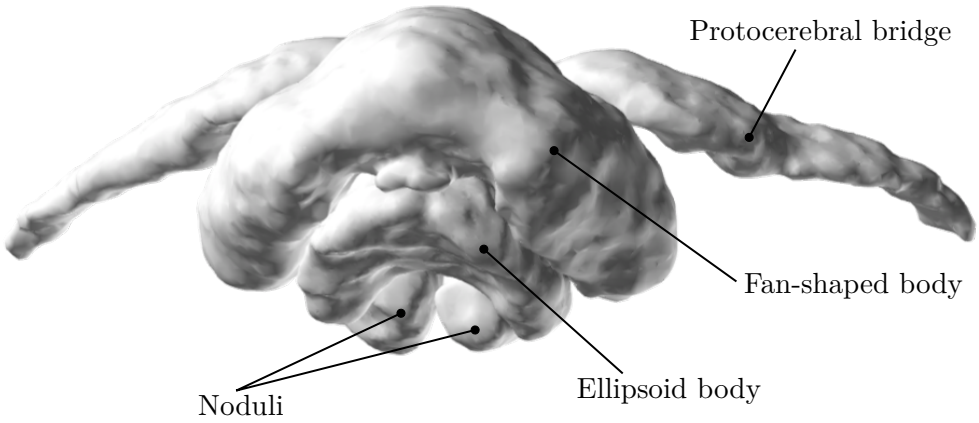

Figure 4.2: Reconstruction of the central complex of a female *Megalopta genalis* with its four parts, the protocerebral bridge, the fan-shaped body, the ellipsoid body and the paired noduli. Image taken from insectbraindb.org, based on data from Stone et al. [2017], with added labels.

Stone et al. [2017] proposed a complete physiological model able to reproduce the biological path integration phenomena for the sweat bee *Megalopta genalis*. This section should give a rough overview on the basic principle of the model, especially the important parts which are necessary to understand the neuromorphic implementation. The reference for this section is therefore Stone et al. [2017] and is not referencing at each point individually.

Providing input to the two noduli, the tangential noduli (TN) neurons are responsible for the optical flow information. Two of them are necessary to encode the speed measurement serving as an odometer, one for each side (left and right). As already mentioned, how the raw optical information influences the TN neurons is still an unknown mechanism.

Performing path integration also needs information about the current head direction, which is provided by a compass network subdivided into three stages. So-called tangential lower division of the central body (TL) neurons collect orientational information outside of the central complex. The central body lower in this case is referred to the ellipsoid body. TL neurons project their orientational input to the columnar lower division of the central body, type 1 (CL1) neurons. As visible in figure 4.2, the ellipsoid body does not have a separation into two hemispheres. It appears that spatial information merges from different sides of the brain outside of the central complex. Last but not least, the CL1 neurons project to the inhibitory tangential protocerebral bridge, type 1 (TB1) neurons. These are located in the protocerebral bridge and are well separated in both hemispheres and each compartment represents a specific heading direction. Both halves of the protocerebral bridge reflects the compass information from CL1 neurons.

Compass information from the TB1 neurons and optical flow information from the TN neurons are projected into the columnar protocerebral bridge/upper division of the central body, type 4 (CPU4) neurons that stretch through the entire central complex. With both important informations available for path integration, it is assumed that the CPU4 neurons encode the information about the nest position.

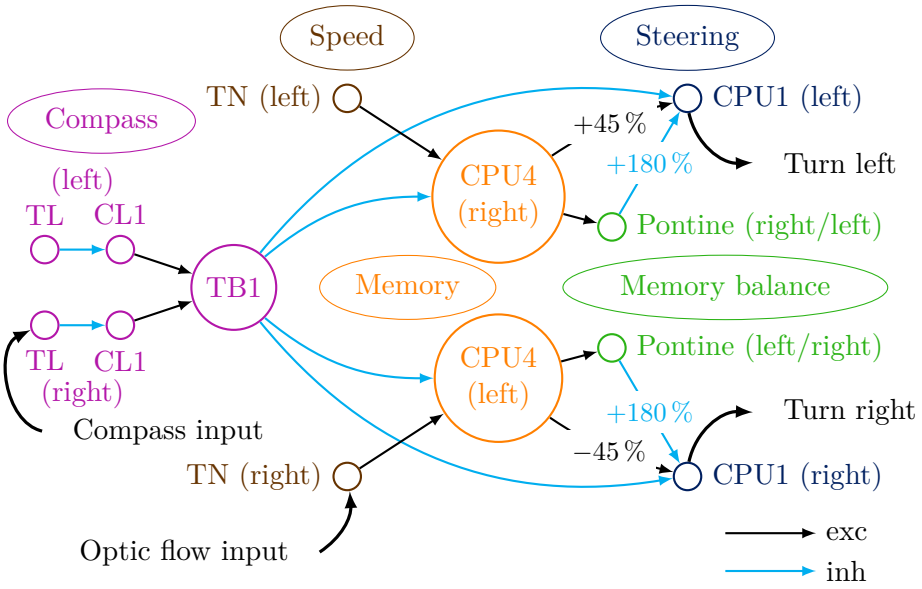

Figure 4.3: Simplified visualization of the path integration network. Black lines denote excitatory, blue lines denote inhibitory synaptic connections. Each circle, besides the TN neurons, describes a subpopulation of 8 neurons each. Every population is assigned to a different compass direction separated by 45° (except for the CPU1 neurons). The connections visible always connect populations with the same orientation. This is not the case for the CPU4 to CPU1 connections with a shift of 45 % clockwise or counterclockwise, depending on the hemisphere and the connection of pontine to CPU1 with a shift of 180 %. Different colors indicate the different tasks of the populations.

Located in the fan-shaped body, the columnar protocerebral bridge/upper division of the central body, type 1 (CPU1) neurons receive input from CPU4 neurons and also collect compass information from the TB1 neurons. Their information converges into brain regions responsible for premotoric control. The summed activity of the CPU1 neurons in each hemisphere is therefore taken as steering signal.

Figure 4.3 shows the network with reduced complexity with an abstract representation of the model. It has to be mentioned that the TL, CL1, TB1, CPU4 and pontine populations are split into eight subpopulations which are assigned to unique directions. Also the CPU1 population is split into eight subpopulations, which are not directly assigned to a certain compass direction because for the steering direction just the hemisphere matters. The TN neurons are only split into left and right hemisphere as they deliver odometric information.

Starting from the compass information, the 16 TL cells (eight per hemisphere) inhibit on CL1 neurons of the same compass direction. The compass direction is preserved for the connection between CL1 and TB1, with the latter receiving two inputs per subpopulation. Since the information of the TB1 is the same for the two protocerebral bridge compartments, the effective vector information is like one population of TB1 neurons as shown in the figure.

The TB1 neurons now inhibit the lateral CPU4 and CPU1 populations with the

compass direction of each neuron is kept. Each subpopulation of the CPU4 neurons receives further input from the TN neurons, which is the same for all in each hemisphere. Excitatory output of the CPU4 population now goes to the pontine neurons with preserved compass direction, while the connection to the CPU1 population is slightly different. The connection of the neurons is cyclically permuted by one angular resolution (45 %), clockwise, or counterclockwise, depending on the hemisphere. This enables the comparison (in the CPU1 neurons) of the stored nest direction (in the CPU4 neurons) to the current head direction.

This can be illustrated with an example. If the head direction is off by one angular resolution from the target, on one hemisphere the inhibitory TB1 signals will cancel out the stronger activation from the CPU4 neurons. In the same directional population on the other hemisphere, the activation from the CPU4 population is however not canceled out by the TB1 signals, resulting in a higher CPU1 activity compared to the other hemisphere. Thus the summed up activity of the whole CPU1 population marks a steering signal for returning home. The whole system is further stabilized by the inhibitory connections of the pontine neurons to the CPU1 neurons, at which the compass direction is rotated by 180°.

To simulate the described network, different assumptions have to be made and put into formulas. First of all, a simple firing rate model was used for each neuron by Stone et al. [2017]. Thus, the output firing rate r_j of neuron j is described as a sigmoid function

$$r_j = \frac{1}{\left(1 + \mathrm{e}^{-(a \cdot I_j - b)}\right)}. \tag{4.1}$$

This rate is dimensionless and can only take values between 0 and 1. The synaptic input I_j is the weighted sum of all neurons i acting on neuron j

$$I_j = \sum_i w_{ij} \cdot r_i. \tag{4.2}$$

Here w_{ij} describes the synaptic weight connecting neuron i with neuron j. These weights only take values of 0 (no connection), 1 (excitatory) and -1 (inhibitory). The parameters a and b are individually tuned for every neuron population. Equation 4.1 is universal for all neurons in the model, while the input I_j is different for the TN, TL and CPU4 neurons. For this reason, these will now be described in more detail.

The physical state of the insect has five degrees of freedom, the head orientation ϕ, the position $\vec{x}$ and the velocity $\vec{v}$, with $\vec{x}$ and $\vec{v}$ being two-dimensional vectors. It makes sense to switch to polar coordinates for the velocity, thus depending on the direction of movement Θ and the absolute velocity v

$$\vec{v} = v \cdot \begin{pmatrix} \cos(\Theta) \\ \sin(\Theta) \end{pmatrix}. \tag{4.3}$$

The difference of ϕ and Θ is visualized in figure 4.4. It shows the difference between head direction and direction of movement. Both angles can be different because of effects like wind and thus have to be considered differently in the model.

According to Schreiber [2021], the internal states of the TN cells given in Stone et al. [2017] had some minor mistake in the written equations. Thus, the internal

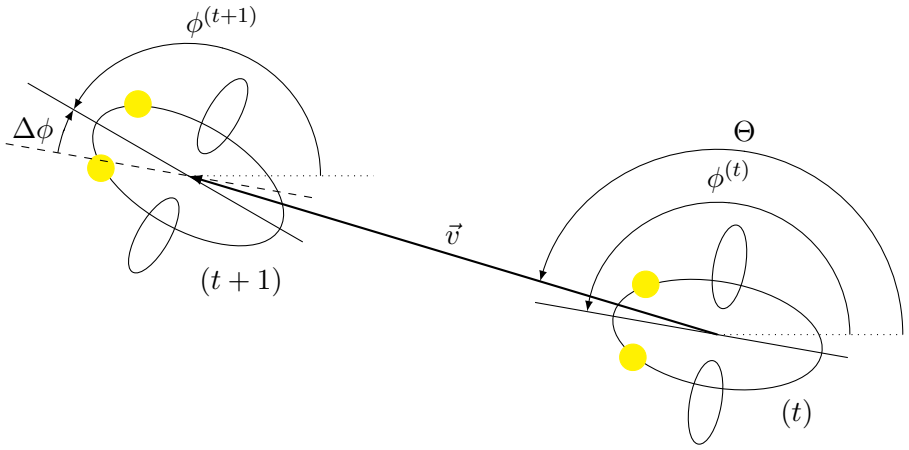

Figure 4.4: Visualization of the insect coordinates. The travelled distance of the insect from time t to $t+1$ is equal to the velocity vector $\vec{v}$, with Θ being the direction of flight. The angles $\phi^{(t)}$ and $\phi^{(t+1)}$ describe the head direction of the current and the following time step, with their difference $\Delta\phi$. The head direction is not necessary equal to the direction of flight.

states were derived by Schreiber [2021] as

$$I_{\mathrm{TN},L/R} = \pm v \cdot \sin\left(\Theta - \phi \pm \phi_{\mathrm{TN}}\right) \mp \rho \cdot \dot{\phi}. \tag{4.4}$$

In this case, ϕ_{TN} can be called sensitivity angle of the TN neurons. Depending on this angle, the direction is determined which evokes the biggest response in the TN neurons. Flying orthogonally to this direction, the response however would be minimal. For the model $\phi_{\mathrm{TN}} = \pi/2$ was used, thus delivering

$$I_{\mathrm{TN},L/R} = v \cdot \cos\left(\Theta - \phi\right) \mp \rho \cdot \dot{\phi}. \tag{4.5}$$

Here, ρ is a constant and proportional to the distance between the two eyes and the change of the head direction is described by $\dot{\phi}$.

As already mentioned, also the TL compass neurons implement their own internal state. With a population of eight neurons, which are separated by $\pi/4$ in angular representation, their output activity encodes the head orientation (neurons indexed by $j = \{0, ..., 7\}$):

$$I_{\mathrm{TL},L/R,j} = \sin\left(\phi + \frac{j \cdot \pi}{4}\right). \tag{4.6}$$

The according output rate can be determined with equation 4.1 passed to the TB1 neurons. As already mentioned, each subpopulation receives input from two CL1 neurons. However, the TB1 neurons also inhibit their own neighbors, thus the internal state is given by

$$I_{\mathrm{TB1},j} = \frac{2}{3} \cdot r^{(t+1)}_{\mathrm{CL1},j} + \frac{1}{3} \cdot \sum_{i=0}^{7} w_{ij} \cdot r^{(t)}_{\mathrm{TB1},i}. \tag{4.7}$$

The inhibitory connections to other cells in the same population functionally imple-

ments a ring-attractor network. There is no self inhibition, so $w_{ij} = 0$ for $i = j$. The weights between other subpopulations are determined by their represented direction. For opposite neurons the inhibition is maximal with $w_{ij} = -1$ for $i = (j + 4)\,\%4$. This causes weakly activated cells to become even weaker, resulting in highly active cells to be even more active due to more inactive inputs.

For the CPU4 integrator neurons the state is given by

$$I^{(t+1)}_{\mathrm{CPU4},L/R,i} = I^{(t)}_{\mathrm{CPU4},L/R,i} + h \cdot \left(r^{(t+1)}_{\mathrm{TN},L/R} - r^{(t+1)}_{\mathrm{TB1},i} - k \right). \tag{4.8}$$

So the CPU4 neurons receive input from TB1 and TN neurons and also depend on their own previous state. The constants h and k define the coupling strength and the damping.

The output frequency of the CPU1 neurons can be calculated with equation 4.1, with the internal state given by equation 4.2. Their influence on the motoric output is however calculated from the sum of each hemispherical population:

$$\Delta\phi = \mu \cdot \left(\sum_i r_{\mathrm{CPU1},R,i} - \sum_i r_{\mathrm{CPU1},L,i} \right). \tag{4.9}$$

So the change in the head direction is determined by the difference of the summed activities in each half. The parameter μ is a heuristically chosen scaling constant.

4.3. Hardware model

As already mentioned, the experiment was already implemented on HICANN-DLSv2 by Schreiber [2021]. Therefore, the whole network size was reduced to fit on the previous prototype with only 32 neurons available. Also some simplifications were made and the neurons had to be calibrated for being usable. With the calibration being one of the most challenging tasks, the goal was to develop a flexible calibration that the experiment can be easily deployed on different chips without human fine tuning. This was not the case on the former implementation. So based on the work of Schreiber [2021], the network, which will be presented in the following, was reimplemented on HICANN-X also using some of its new features.

Within the model of Stone et al. [2017] spikerates were used to model the neuron dynamics. Thus, using LIF based neurons and connecting them with each other is a plausible approach to implement the model. The LIF firing rates must first be normalized to the interval of zero to one because spikes are just discrete values. With the amount of spikes occurring in a certain time interval, the spikerate can be determined. This spikerate depends on the inputs and furthermore the LIF parameters, such as the gain of the synaptic input OTA. With a minimum rate of zero spikes, the lowest possible spikerate is naturally given. For the maximum rate Schreiber [2021] used a rate of $r_{\max} = 100\,\mathrm{kHz}$. This was motivated by the observation of rates of 150 Hz in *Megalopta genalis* Stone et al. [2017] in biology, thus with a speedup factor of approximately 1000, a hardware rate of 100 kHz is plausible. Of course also a different rate can be chosen for normalization.

The network implemented on the hardware is visualized in figure 4.5. First of all, the TL and CL1 populations for the compass direction are completely left out and only TB1 neurons with four neurons per population instead of eight are used.

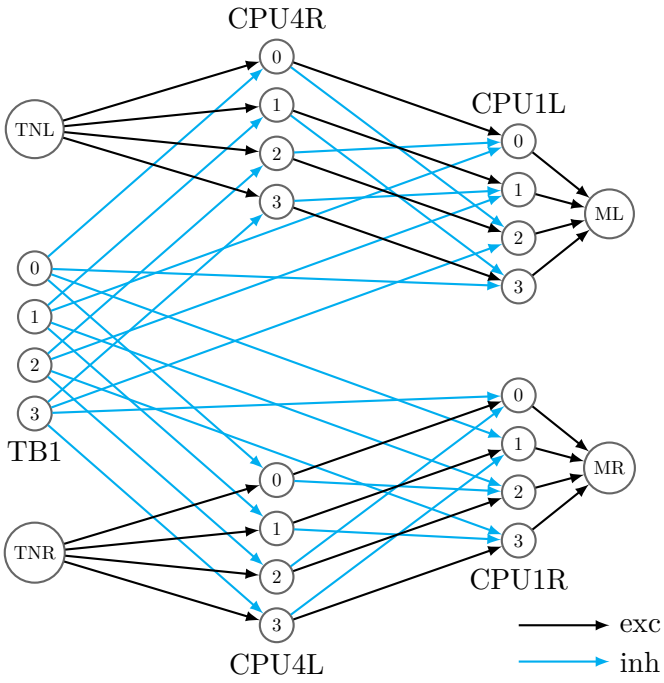

Figure 4.5: Hardware network implemented on HICANN-X. The TB1 and TN population are marking the input of the network and are fully virtually simulated. Acting on CPU4 and CPU1 populations, which are implemented on the analog core, the output signals of the CPU1 neurons are added on separate motor neurons responsible for the steering signal. Each population just contains four neurons instead of eight. Black arrows describe excitatory, blue arrows inhibitory synaptic connections.

This does not represent a problem, because the whole compass information is also contained in the TB1 population. They are not implemented as hardware neurons on the analog core, instead the compass signals are generated by the background spike generators introduced on HICANN-X. The rate output on hardware can therefore be calculated with

$$r_{\mathrm{TB1},j} = \frac{r_{\max}}{2} \cdot \left(1 + \sin\left(\phi + \frac{j \cdot \pi}{2}\right)\right), \tag{4.10}$$

which is in line with equation 4.6. The four neurons are indexed with $j = \{0, ..., 3\}$. With eight spike sources in total even the full model using eight directions could be implemented.

In the model on hardware it was further assumed that the head direction is always identical with the direction of flight, i.e. $\phi = \Theta$. Thus from equation 4.5 follows

$$r_{\mathrm{TN},L/R} = r_{\max} \cdot \left(\frac{v}{v_{\max}} \mp \rho \cdot \dot{\phi}\right), \tag{4.11}$$

with $\rho = 0.614$ set for the hardware experiment. Also the velocity was assumed to be constant for the whole experimental run, specifically $v/v_{\max} = 0.5$ was used.

Implementing the CPU4 dynamics given by equation 4.8 is more challenging. It shows that each state also depends on its previous state, thus the path integration time constants are in the order of seconds, or minutes in the biological time domain. On HICANN-X the neurons can be configured for time constants in the range of up to approximately 100 µs. This is, however, not sufficient to implement the required integration dynamics. For this reason an extraneural mechanism is considered, which operates on the synapses. These store according to their weight the home vector. So it is desired to have a wide weight-range, not given by a synaptic weight of 63 at maximum. If the precision should be high, then the maximal accessible spatial range is low for such a low dynamic weight range and vice versa. Therefore 16 synapses were merged together per CPU4 neuron to a "supersynapse". This allows roughly a 10-bit integer range from 0 to $1008 = 16 \cdot 63$ for the weights. Based on equation 4.8, the weights $w_{\mathrm{CPU4},L/R,i}$ are updated according to the rates of the TB1 and TN neurons with

$$w^{(t+1)}_{\mathrm{CPU4},L/R,i} = w^{(t)}_{\mathrm{CPU4},L/R,i} + \frac{h}{r_{\max}} \cdot \left(r^{(t+1)}_{\mathrm{TN},L/R} - r^{(t+1)}_{\mathrm{TB1},i} - k \right). \tag{4.12}$$

These weights have to be transformed into spikes, by means of a constant background source activating the supersynapses with a rate of 100 kHz. With a weight of 1008 the output rate should be 100 kHz, for a weight of 0 of course also the output rate should be 0 kHz with a linear behavior between these weights. The calibration is further described in section 4.4.1. With this implementation the connections to the CPU4 neurons in figure 4.5 excite or inhibit the neurons just by weight changes. Integration starts with a weight of 504 being in the middle of the weight range with an approximate output frequency of 50 kHz.

In the model of Stone et al. [2017] the excitatory connection between CPU4 and CPU1 is shifted by the neighboring compass direction clockwise or counterclockwise, respectively. This shift is not implemented in the hardware model. Instead the inhibitory TB1 connection is shifted to get the same effect. At the end it does not matter which connection is shifted, because the CPU1 neurons are not assigned to a specific compass direction and all rates on one hemisphere are summed up at the end. Without the restriction of Dale's principle [Strata and Harvey, 1999] on hardware, describing that the synaptic connections of a neuron to other cells performs the same chemical action, the pontine neurons could be left out. They just had the same output rate as input rate, but inhibited on the CPU1 neurons shifted by 180°. Therefore the inhibitory connection from CPU4 to CPU1 is shifted by 180°.

For the CPU1 neurons one should use a sigmoid function as given by equation 4.1. This is however not easily implementable on the hardware, so Schreiber [2021] used another target response:

$$r_{\mathrm{out}} = r_{\mathrm{exc}} \cdot \left(1 - c \cdot \frac{r_{\mathrm{inh}}}{r_{\max}} \right). \tag{4.13}$$

With a factor $c = 0.5$ the output rate would be 50 kHz when both inputs fire with 100 kHz. The output rate also increases linearly with the excitatory rate for a fixed inhibitory input and vice versa. Most crucial is the response at the operating point in the middle of the sigmoid function at $x = 0$. In this area the sigmoid function can be approximated to be linear and with the summed activities on the left and right

subpopulations, so being not sigmoid should not effect the network's behavior. The calibration algorithm is further described in section 4.4.2.

Last but not least, the motor neurons have to be considered. Receiving excitatory input from all CPU1 neurons of the same hemisphere, they are responsible for the summation of the individual CPU1 rates. So the steering signals is calculated from the output rates of the two motor neurons as

$$\Delta\phi = \Delta\Theta = \frac{\mu}{r_{\max}} \cdot (r_{\mathrm{M},L} - r_{\mathrm{M},R}) \,, \tag{4.14}$$

with the parameter $\mu = 0.982$. Without inhibitory input signals the same calibration as for the CPU1 neurons can be used, because a linear dependency on the input rate can also be used to sum these up.

4.4. Calibration

We developed special calibration routines for the neurons involved in the insect experiment based on the calibration algorithms presented in chapter 3. To do so, the gain of the synaptic input OTA and the synaptic time constant were fine tuned after an initial calibration of the LIF parameters. The aim was to replicate the neuron responses specified in the model presented above. Furthermore, the calibration should be flexible and run without human influence on different chips. From now on the calibration of CPU4 neurons is referred to as a weight-rate calibration, because the output rate depends on weights: rate (weight). The same applies for the CPU1 calibration being called rate-rate calibration with the output rate depending on different input rates: rate (rate). Each neuron involved in the experiment is calibrated individually instead of all neurons at once. This is done due to a bug found in the synapses, the spike's amplitudes differ if signals are send to only one or all neurons. A detailed description of the problem is given in appendix A.5.

4.4.1. Integrating neurons

To get an output rate depending on the weight, the synapses of the weight-rate neurons are activated with a rate of 100 kHz. With currently only 4 of the 8 background spike sources active, one of the remaining sources could be used for this. However, since the full model would require 8 compass directions, it would use all background spike sources. Equation 4.10 shows that the rates of opposing compass direction neurons always sum up to 100 kHz. Therefore, two opposing compass directions can be used to generate the input of the CPU4 neurons. These two spike sources are also connected to different CPU1 neurons. So they have to be distinguishable and cannot fire to the same synapse to excite the CPU4 neurons. As a result, two synapses are configured with an equal weight, each receiving spikes from a single spike source. With this method 32 synapses per neuron have to be allocated instead of 16.

Before the precise calibration procedure starts, the LIF parameters of all neurons are calibrated with the routines presented in chapter 3. The neuron properties in

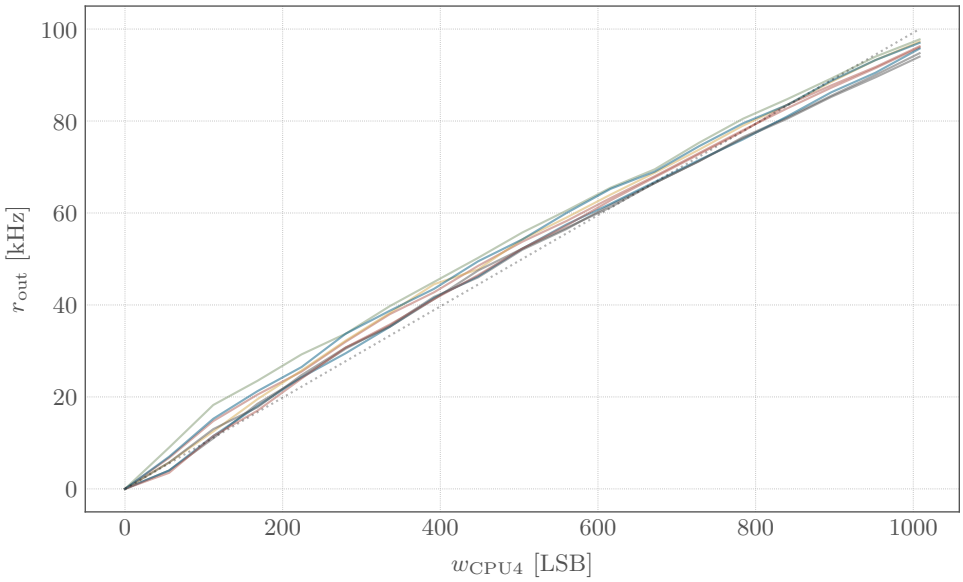

Figure 4.6: Weight-rate calibration of 8 neurons in quadrant 0 on chip 22 on setup 63. The output rate of each neuron depends on the total weight. Indicated by the dotted line is the desired target curve. This target is closely reached by the calibrated cuves, while for weights below 600 the output rates are a little bit higher, a rate of 100 kHz is closely not reached with the highest weight of 1008.

SI-units were heuristically chosen, to obtain optimal results.

$$V_{\text{leak}} = 0.50\,\text{V} \qquad \tau_{\text{syn, exc}} = 8\,\mu\text{s} \qquad \frac{C_{\text{mem}}}{g_{\text{syn, exc}}} = 1\,\mu\text{s}$$

$$V_{\text{reset}} = 0.20\,\text{V} \qquad \tau_{\text{syn, inh}} = 8\,\mu\text{s} \qquad \frac{C_{\text{mem}}}{g_{\text{syn, inh}}} = 1\,\mu\text{s}$$

$$V_{\text{thresh}} = 0.60\,\text{V} \qquad \tau_{\text{mem}} = 60\,\mu\text{s} \qquad \tau_{\text{refr}} = 0.32\,\mu\text{s}$$

For the weight-rate calibration, the inhibitory input is not required as the synapses just receive excitatory spikes. Therefore, the inhibitory input is completely disabled for this type of neurons. Its synaptic time constant and the gain of its OTA is still calibrated initially, because these values also serve as a basis for the rate-rate calibration presented later.

For the calibration the excitatory time constant was dimensioned close to the period of the input frequency of 100 kHz, which is 10 µs. This benefits the calibration as the synaptic input line potential will be always stimulated, resulting in a constant current onto the membrane. The resulting output frequency can afterwards be scaled with the transconductance value of the input OTA. A output rate of 75 kHz was targeted with a weight of 756 scaled by $g_{\text{syn, exc}}$.

Figure 4.6 shows the resulting calibration of 8 neurons in quadrant 0. These were randomly selected and later used in the experiment. In principle every neuron can be used for the experiment, because cherry-picking was not necessary. Also every

neuron closely matches the target, indicated by the dotted curve. The experiment starts with a weight of 504, at this point a rate of 50 kHz is desired as this rate marks the equilibrium. However, the rate for all neurons is a little bit higher at his point lying around 54 kHz.

Unfortunately the small deviations from the target behavior could not be mitigated. That is because the output response of LIF-neurons follow the frequency-current (f-I) relationship [Bogaard et al., 2009], which is not described by a linear function. A stronger leak for example (smaller V_{leak} or smaller τ_{mem}) will cause an even bigger deviation. This can be explained that for a strong leak in combination with small weights the output rate is zero, because the threshold is not reached. At a certain weight, the output rate starts to increase more and more in this scenario. Shifting the curve with the synaptic input bias, that at the maximum weight also the maximum rate is reached, will result further deviations. Also a bigger refractory time limits the output rate, thus it is set to a small value. The same effect was observed for smaller synaptic time constants.

4.4.2. Steering neurons

In contrast to the weight-rate calibration, the CPU1 neurons do not rely on synaptic modulation. They receive one excitatory and two inhibitory spike inputs. So again some initial LIF parameters are chosen heuristically in SI-units.

$$V_{\text{leak}} = 0.50\,\text{V} \qquad \tau_{\text{syn, exc}} = 8\,\mu\text{s} \qquad \frac{C_{\text{mem}}}{g_{\text{syn, exc}}} = 1\,\mu\text{s}$$

$$V_{\text{reset}} = 0.20\,\text{V} \qquad \tau_{\text{syn, inh}} = 8\,\mu\text{s} \qquad \frac{C_{\text{mem}}}{g_{\text{syn, inh}}} = 1\,\mu\text{s}$$

$$V_{\text{thresh}} = 0.85\,\text{V} \qquad \tau_{\text{mem}} = 60\,\mu\text{s} \qquad \tau_{\text{refr}} = 0.8\,\mu\text{s}$$

These starting values except for V_{thresh} and τ_{refr} are the same as for the weight-rate starting values. This time of course, the inhibitory input is of importance.

Analog to the weight-rate calibration routine the synaptic time constants are dimensioned as two long values. Again this should ensure a constant current from both synaptic OTAs to the membrane. Scaling the transconductance values to desired output rates splits the calibration into two parts:

1. Expect an output of 100 kHz with an excitatory rate of 100 kHz and no inhibitory input by adjusting $g_{\text{syn, exc}}$.

2. $g_{\text{syn, inh}}$ is tuned to get an output of 50 kHz with both inputs firing at the maximum frequency of 100 kHz.

Figures 4.7 and 4.8 are showing the result of the calibration. Like for the weight-rate calibration, the 8 neurons are randomly selected from quadrant 1. Indicated by the dashed line, the working point with an output frequency of 50 kHz is well defined. The individual deviations are similar for all neurons, thus the inter-neuron deviation is less significant compared to the deviation of the target result. Higher rates than expected appeared for a low inhibitory input. However, the input rates of the CPU1 neurons only seldomly fall in this range. A low inhibitory input from CPU4 neurons implies also a high excitatory input from the CPU4 compass direction

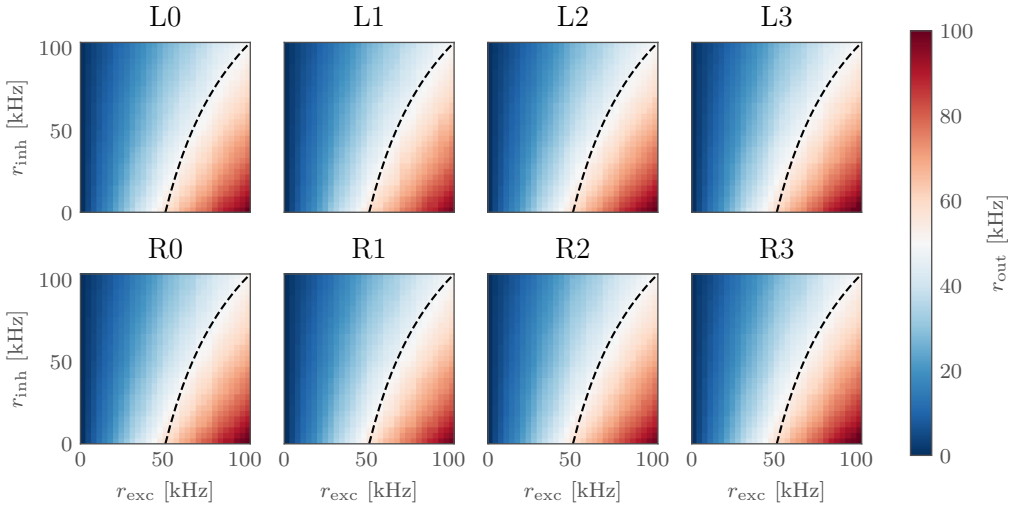

Figure 4.7: Resulting output rate of the CPU1 neurons with different excitatory and inhibitory input. The dashed line marks the region where a rate of 50 kHz is expected. 8 random selected neurons on quadrant 1 measured on chip 22 on setup 63.

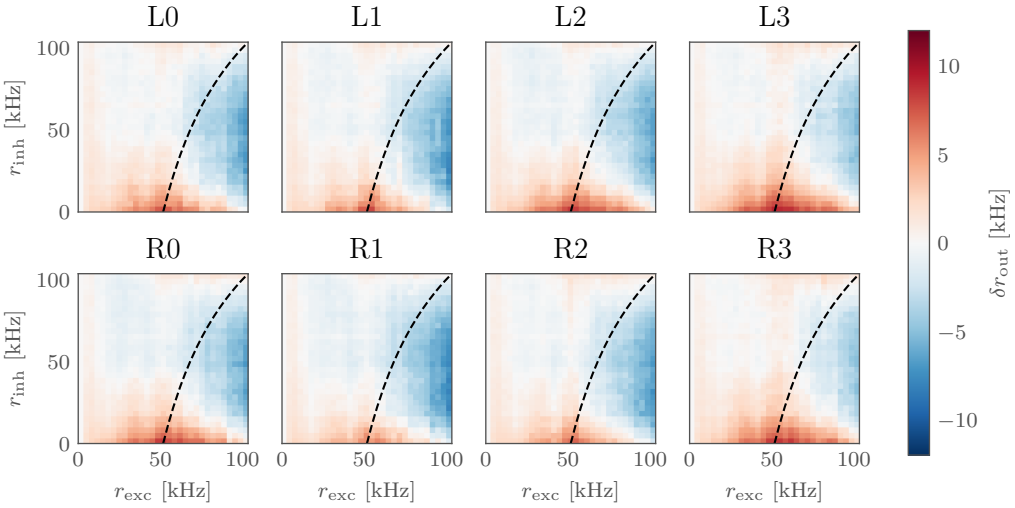

Figure 4.8: Deviation of the calibrated output rate from the expected rate described by equation 4.13. The dashed line describes the region with an expected output rate of 50 kHz. All neurons have a similar deviation pattern, the rate is higher for a small inhibitory rate with moderate excitatory rates and smaller for high excitatory rates with a moderate inhibitory input. As all neurons behave slightly similar, so the deviation from the target is more significant. Same neurons used as in figure 4.7.

counterpart. Thus, the output would be rather in the lower right corner, resulting in less deviations. Also the TB1 neurons inhibit the CPU1 neurons, making an inhibitory rate of zero unlikely. In contrast to the deviations for low inhibitory input,

lower rates than expected were observed for high excitatory input rates. With the deviations being low, this should be also no problem. Considering that such high rates only appear for strongly integrated CPU4 neurons, the insect should still find back to its nest and in their equilibrium for excitatory input rates of 50 kHz, the deviations are lower.

The same calibration is also used for the motor neurons to sum up the excitatory rates of every CPU1 neuron per hemisphere. In this case, the inhibitory rate is always zero and just the excitatory input rates are varied. For this setting higher rates as expected are measured, but this should be minor, because the deviations are similar for every neuron and in the end just the difference between the motor neurons is of importance. The steering neurons (CPU1 and motor) also have no influence on the stored home vector. Thus, the information of the travelled path is just contained in the CPU4 neurons. If the steering however is inaccurate, then the nest is still not reached, but the information about its position is kept.

4.5. Experiment

At the beginning of every experiment different parameters are transferred from the host computer onto the PPU. First of all, a random seed is given to determine the random walk at the beginning. Also a total runtime t_{stop} is given. After t_{return} the insect will stop its outbound journey and returns home to its nest. The parameters k (CPU4 decay) and h (CPU4 update scaling) for the CPU4 weight update (equation 4.12) are also chosen at the beginning. All parameters are integer values, because the PPU does not provide hardware support for floating point operations. Parameters like μ from equation 4.14 and ρ from equation 4.11 can still be given as fractional numbers, because the angles on hardware are integers running from 0 to 1023, like it was done by Schreiber [2021]. The sine function is then implemented as a lookup table filled with `int16_t`, the maximum values there represent -1 and 1, respectively. Converting them to `int32_t` for multiplications and further division is done afterwards, resulting in fractional numbers.

Besides transferring some parameters to the PPU, the host computer also configures the chip and loads the calibration. The whole routing is also determined by the host computer and the initial CPU4 weights are set to 504. Afterwards, the whole experiment runs independently on the PPU executing the simulation and generating the sensory input spike sources.

In the current implementation, the background spike sources continuously feed the network with spikes and the involved neurons spike according to their input. With the PPU, the number of spikes in a certain time interval is determined, which gives the spikerate. The state of the network changes when the weights of the CPU4 neurons are updated according to the update rule given by equation 4.12. Currently the whole experiment can be divided into four subcycles:

1. Extract TN and TB1 rates according to the state of the insect and update the background spike generators. Also update position according to the velocity.
   a) If the insect is on its outbound journey, perform the random walk.
   b) If the insect is on its journey to the nest, update the position according to the rates of the motor neurons.

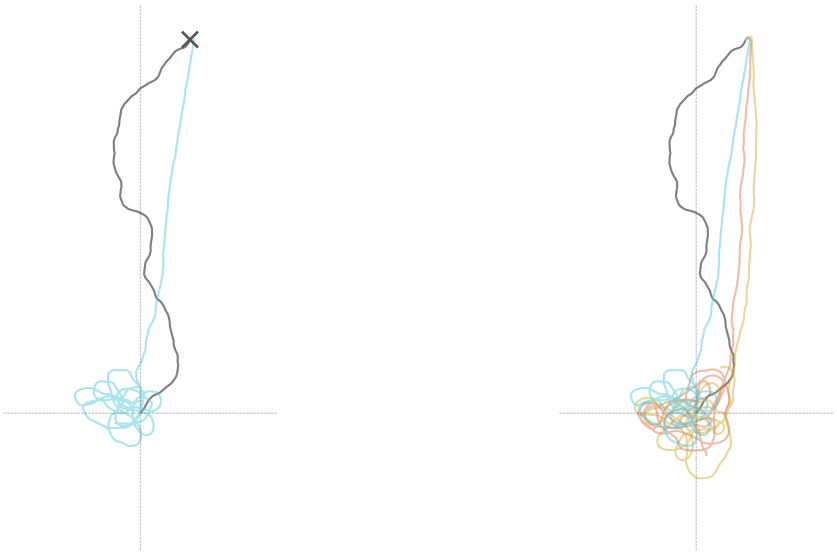

Figure 4.9: Trajectory of a insect with outbound and inbound paths emulated on hardware. Starting from the nest, the outbound path (black) is determined by a random seed. The food location is marked with a X and afterwards the insect returns home (blue). Close to the nest it starts to loop around it. On the right the same seed was given and three different inbound paths from three different setups are shown. The blue path was recorded on chip 22 on setup 63, the red path on chip 30 on setup 69 and the gold path on chip 31 on setup 73.

2. Calculate the update of the CPU4 weights.

3. Update CPU4 weights and the simulation state. Also get the amount of spikes in the spike counters for a certain interval.

4. Save insect position and neuron rates in the FPGA memory.

A total of 2000 steps are chosen for t_{stop}, with the return starting after 500 steps. The total runtime is mostly influenced by the length of the time interval, in which the amount of spikes is registered. Other parts in each step just have a small contribution to the total runtime. But also additional waits can be added to scale down the execution time to real time. Then it would be also possible to use the network in robotic applications with external odometer and compass inputs. That is because the network runs continuously in the background while the update steps are done discretely. In the original implementation from Schreiber [2021] the experiment was executed with 1000-fold acceleration, scaling down the observation interval makes this also possible on HICANN-X. Also the whole 2D environment, simulated with `int16_t` for x and y direction, is calculated faster because of a faster PPU clock speed of 250 MHz compared to 98 MHz on HICANN-DLSv2.

Figure 4.9 shows the trajectory of an insectoid agent performing path integration on hardware. The outbound path in black, starting from the nest, is performed by a random walk predefined by the random input seed. After t_{return} is reached, on

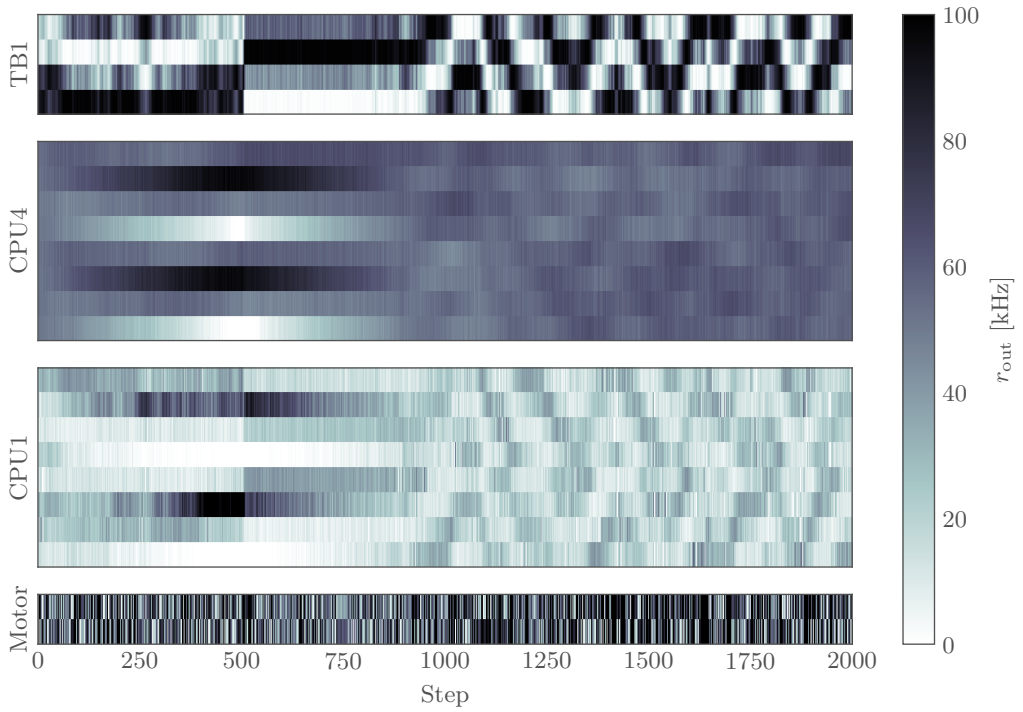

Figure 4.10: Activity of the network, for the trajectory shown in figure 4.9. The spike rate of each neuron and the according step is shown. The shading represents the firing rates (white for a rate of zero and black for 100 kHz as the maximum). It is clearly visible that the CPU4 neurons integrate the already travelled path by moving away from the initial rate of 50 kHz. On the returning journey, they again approach their equilibrium with an output rate of 50 kHz.

hardware after 500 steps, the insect starts to get back based on the motor neuron signals. Also about the same amount of steps are necessary afterwards to come back close to the nest. Within a total of $t_{\text{stop}} = 2000$ steps the simulation stops. As one can see in the figure, at the end the insect loops around the nest. In biology it would find the nest entrance based on other visual cues or scents [Butler et al., 1969]. This is not included in the model and so it steers around the nest position.

On the right in figure 4.9, the inbound path is shown for different setups and chips with the same outbound. Every setup had its own calibration and afterwards the experiment was conducted. It shows that the nest is reached and looped around on every setup. Therefore on can conclude that with the calibration, the experiment runs on every hardware setup independent of manual human interaction.

The network activity for the trajectory on the left side in figure 4.9 of each individual neuron is visible in figure 4.10. During the outbound path, the different compass directions are directly visible. With the path mostly facing a northern direction at the beginning, the according TB1 neuron is expected to be highly active. When the insect moves eastwards at the beginning of the outbound journey an increase of activity of the third neuron is registered, therefore it marks the east direction. According to equation 4.12, the compass signals inhibit the weights of the CPU4

neurons. Therefore, the northern CPU4 neurons spike less and less, while the rates for the southern CPU4 neurons increase. Unfortunately, the odometric influence of the TN neurons on each CPU4 hemisphere can only hardly be recognized in the visualization.

After 500 steps, the insect starts to move according to its motor signals, which are represented by the summed CPU1 rates. It is visible that the first four neurons of all CPU1 neurons have a smaller rate if summed up compared to the other four neurons at this point. The latter belong to the lower hemisphere representing the population for the steering signal to the right. Thus, the insect moves to the right according to equation 4.9. Afterwards the path is a straight line in south-west direction, the northern TB1 activity changed to a low rate. Also the different CPU4 neurons start to approach to their initial rate of 50 kHz. The CPU1 sum is the same for both hemispheres, this explains the straight path to the nest direction.

After approximately 900 steps, the closer surrounding of the nest is reached and the insect starts to loop around it. Compass rates are always changing with the head direction. The rates of the CPU4 neurons however now remain around 50 kHz. According to the compass input, the rates of the steering neurons also change constantly. Therefore the motor signals always correct the current path resulting in these loops.

To perform the experiment as described above, also the parameters h and k from equation 4.12 need to be adjusted. So these parameters were sweeped to determine the best parameter set for a successful experiment. The result of the sweep is illustrated in figures 4.11 and 4.12. Figure 4.11 gives a more intuitive visualization of the sweep and the influences of the different parameters, while figure 4.12 gives a more qualitative description.

With the CPU4 update scaling h, the sensitivity of the integrator is controlled. If the set value is too small, a lower precision of the homing vector is to be expected. As a result, the data points are further away from the nest, which can be seen in figure 4.11 and also on the top right of figure 4.12. Also the radii of each individual loops is much higher. The advantage of lower values however is that clipping does not occur, which is a big problem for higher values of h. Clipping occurs if the lowest or highest possible weight, respectively, is reached during the random walk. In this case the home vector is shifted because path information is lost, so the insect does not find back to the true location of its nest. These outliers can be seen in figure 4.11 with trace points at the end of the box. Also the qualitative view from figure 4.12 shows that the different trajectories differ from each other. Due to the stronger response even for smaller changes, the standard deviation of individual loops is smaller. The looped center however differs from the true location of the nest after clipping.

The CPU4 update scaling k however serves as damping constant for the CPU4 weights. Positive values for k ensure that the overall CPU4 weights go down by time, while negative values do the opposite according to equation 4.12. Figure 4.11 suggests that for positive values of k the individual loops are bigger compared to the negative values. In figure 4.12 on the bottom right, this observation can be confirmed. This could be due to an overall smaller output rate of the CPU4 neurons with smaller weights by time. The CPU1 neurons have therefore a smaller excitatory input, thus the steering signals are also smaller resulting in bigger loops. Higher

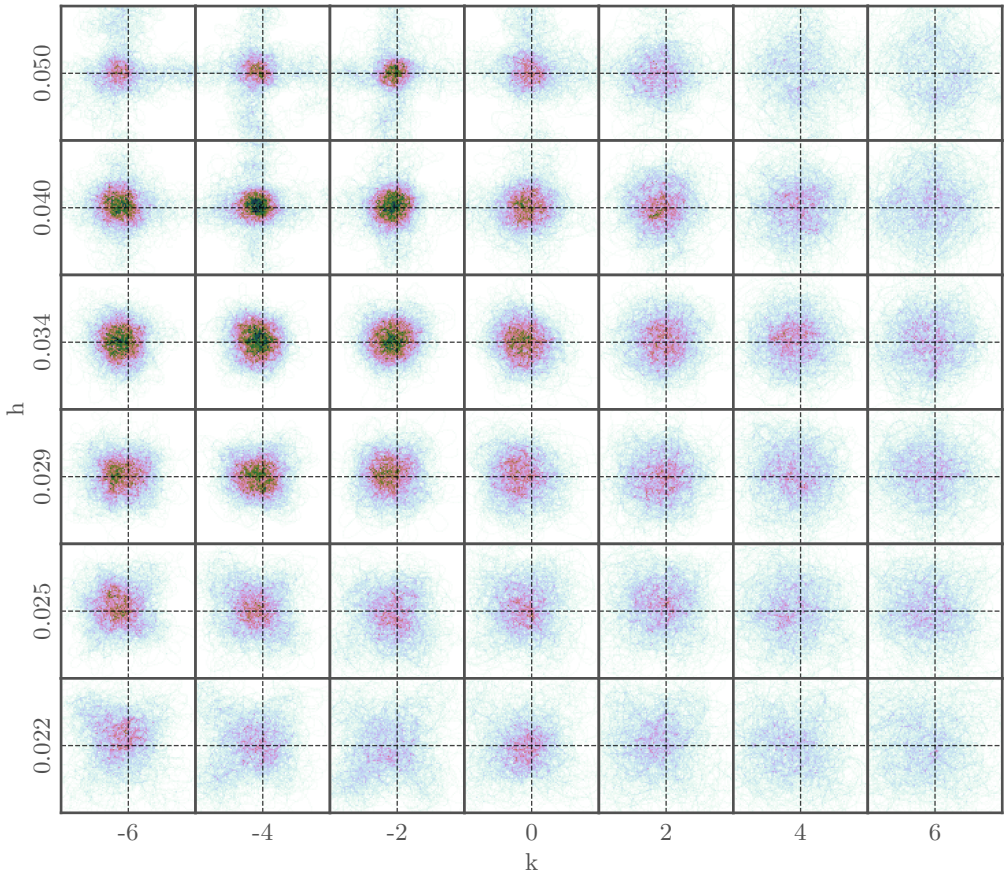

Figure 4.11: Sweep of h and k and their resulting influence on the network performance. Data is based on 100 independent experiments for each parameter set with different starting seeds. In each square only the looping phase for $t > t_{\text{stop}}/2$ is considered in the histogram, with the middle as nest. From the nest, each direction spreads by 6000 (simulation units). The number of samples is encoded by the color from bright to dark.

excitatory input, corresponding to negative values of k, seem to benefit the individual looping radii.

Looking at figure 4.11, a value of $k = -2$ or $k = -4$ combined with a value of $h = 0.034$ seems to be the best choice. All trajectories seem to be located around the center with no outliers with these combinations. Also the individual looping radii seem to be small. This estimation is confirmed by figure 4.12. The mean distance from the origin and also the standard deviation across different loops is smallest for $h = 0.034$ and $k = -4$. Also like shown on the bottom right each individual loop has a small standard deviation, thus finding the true location of the nest is done best with this parameter set.

With this parameter set also the trajectories from figure 4.9 were recorded. It was also used for figure 4.13, showing the outbound and inbound trajectories for 30 different experiments.

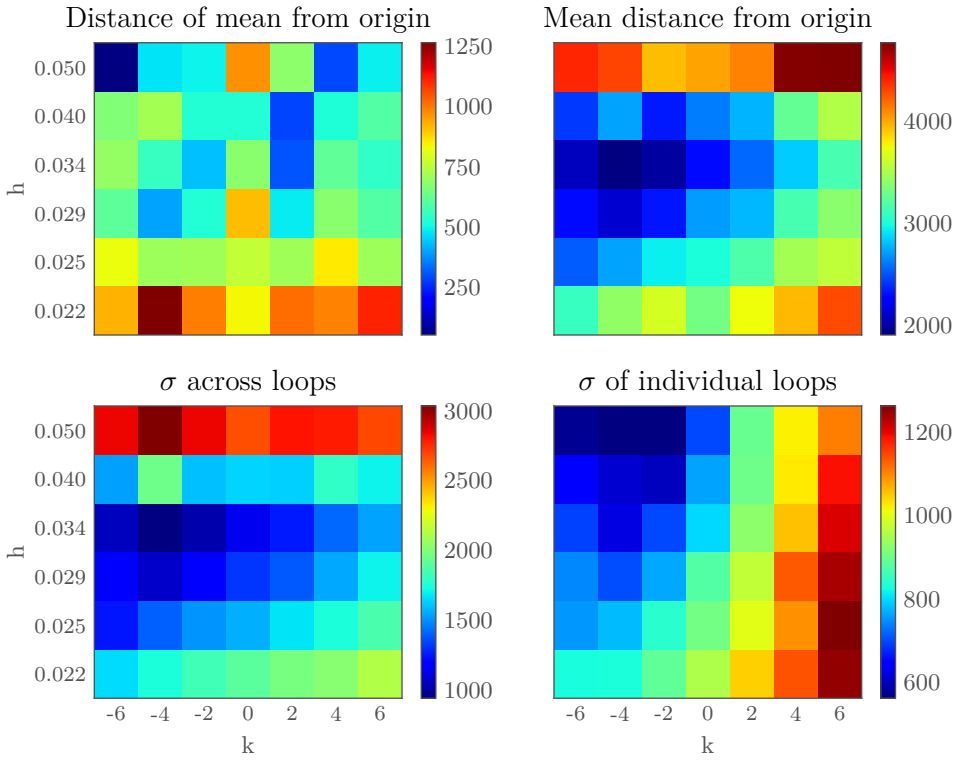

Figure 4.12: Sweep of h and k and their resulting influence on the network performance. Data is based on 100 independent experiments for each parameter set with different starting seeds. Only the looping phase for $t > t_{\text{stop}}/2$ is considered. In the top left plot, the distance of the mean from the nest is visible. The plot on the top right shows the mean distance from the nest of all curve points. Higher values indicate that there are points which are far away from the nest. On the bottom left the standard deviation of the looping radii for all data points is plotted. This indicates whether the insect returns to the same point repeatedly, or if it scatters. Last but not least, the plot on the bottom right shows the standard deviation of each single looping radii. Small values indicate tight individual loops. The units correspond to the simulation unit on the hardware.

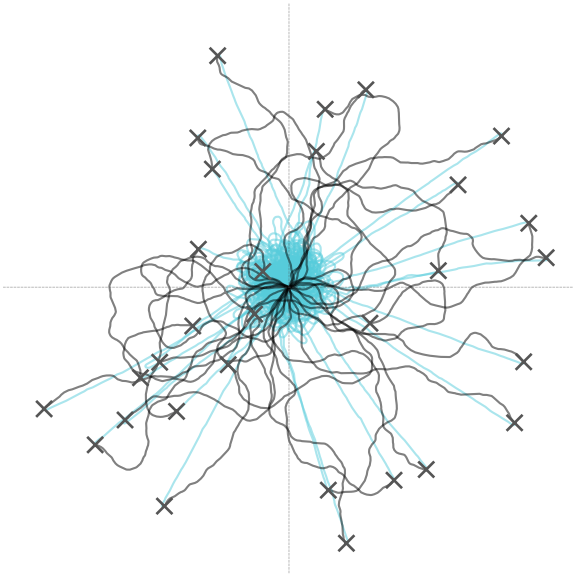

Figure 4.13: Outbound and inbound trajectories of 30 different experiments with different seed. All return trajectories are looping around the nest. Data taken from chip 22 on setup 63.

5. Discussion and Outlook

In this thesis we presented calibration routines fully relying on BrainScaleS-2's embedded processor (PPU). For this purpose we only accessed observables available to the PPUs, including spike rates via the spike counters and voltages via the CADCs. Utilizing the developed calibration routines, we ported a closed-loop experiment to the latest generation of BrainScaleS-2 chips. In particular, we emulated an agent steered by a neural, insect-inspired path integration circuit on its return from foraging trips.

Calibrating the CADCs, the standard deviation between the individual channels was brought down below 1 LSB, which corresponds to 4.5 mV. The whole routine could be executed within 0.5 s. After calibration the CADCs were used to in turn calibrate the different potentials parameterizing the LIF neuron model. It was possible to reduce the individual mismatch across the neurons of an ASIC. Exemplary for the threshold, the standard deviation was brought to about 2 % of the mean value. Also different target values in SI-units can be selected, the threshold is then adjusted to be within 5 % of the target. For each voltage parameter, the calibration takes approximately 0.3 s for the whole chip.

Other technical parameters and the remaining LIF-parameters, including the membrane time constant and the two synaptic time constants, were also calibrated. Different approaches were discussed to estimate these quantities from the limited observables available to the PPU. The results were verified with host-based methods. Calibrating the membrane time constant took approximately 4.5 s and yielded a standard deviation of 2 %. Different target values can be flexible reached, the fact that only minor systematic deviations could be observed validated our approach. Taking about 0.5 s, the calibration for the synaptic time constants can reduce the individual mismatch between neurons to below 5 %. With an accuracy of about 7 %, different target values can be flexibly reached. Since these calibration routines depend on previously calibrated quantities inaccuracies from error-propagation are expected.

For the closed-loop experiment, we further developed model-specific calibration routines to tune the neurons towards a desired high-level behaviour. The model of insect path integration made it possible for an insectoid agent to return to its nest after a flyout phase simulated by a random walk. This closed-loop experiment was completely implemented on-chip, the network of neurons was emulated by the analog core, while simulating the surrounding and trace of the bee was implemented on the PPU. Our calibration algorithms allowed to replicate the experiment on multiple chips and experimental setups.

Currently the whole parameter set of the LIF-neuron can be calibrated to specific target values in less than 30 s. Different methods and optimizations were discussed to even further reduce the total runtime. The presented algorithms can also be used to calibrate multiple HICANN-X chips in parallel. Since the number of PPUs scale with the chip count, this will be able without increasing the overall runtime.

Currently the implemented model for path integration is restricted to four neurons per population and some simplifications were made in the compass signal generation. The insect experiment was originally designed for HICANN-DLSv2 which only of-

fered a limited number of neurons, on HICANN-X these restrictions no longer apply. So the original model with eight neurons per population and an extended emulation of the compass directions by including also TL and CL populations could be investigated. Additional components of the original model, e.g. wind, could in the future also be included. Further extensions such as obstacles avoidance could then also be investigated. Since the model can be slowed down to operate at real time, the model could be used to steer a physical robot by incorporating observables from its actual surroundings.

During the implementation of the calibration routines and execution of the insect experiment some shortcomings of the current hardware implementation became apparent and led to first changes for later revisions. Their improvements could facilitate calibration and experiments in the future. An increased range for the leak potential is already under consideration. In addition, a switch to fully disable the leak term was inserted.

Furthermore, an extension of the synaptic array's memory control could allow accessing neuronal spike counters and configuration as well as the CapMem cells via the vector unit. Currently data from the vector unit has to be transmitted to the general purpose part of the PPU in order to adjust or read these parameters. In addition an extended instruction set with unsigned and signed operations would be desirable.

A. Appendix

A.1. Assembly for binary add function

Adding different bits in the binary search is done on the vector unit. The code can be found below using four different vector registers. Firstly, the `var` register stores the currently used CapMem value. Which bit is set is determined by the `bit` register. The `max` register marks the maximum value which should not be exceeded. Last but not least, a `help` register is used to temporarily store some calculations. In this case just the halfword algorithms are described for `int16_t`. The same also works for byte operations, but the instructions have slightly different names.

```
1  "fxvaddhm %[var], %[var], %[bit], 0\n"
2  "fxvsubhm %[help], %[var], %[max], 0\n"
3  "fxvcmph %[help]\n"
4  "fxvsubhm %[help], %[max], %[help], 1\n"
5  "fxvsel %[var], %[var], %[help], 1\n"
```

In the first line, the bit is added to the current values by a modulo addition. As the instructions only operate on signed integers, the halfword mode only allows the iteration of 15 bits and the byte mode only allows 7 bit sweeps. Afterwards, the maximum is subtracted from the gained new parameter and stored in the helper register. This register is now compared to zero and in the vector condition register (VCR) [Friedmann and Pehle, 2020] it is stored, whether is was above, below or equal to zero. If the parameter was bigger than the maximum, the difference is subtracted from the maximum (line 4) and according to the condition, the sweeped parameter is selected in line 5.

A.2. Assembly for binary subtract function

For the subtraction in the binary search a similar algorithm is used. Another register is of importance called `zero`, which is just filled with zeros. Of course also the maximum register `max` was replaced with the minimum register `min`. Before the subtraction is executed, the `decide_func()` stores in the VCR whether the bit should be subtracted or not.

```
1  "fxvsel %[help], %[zero], %[bit], 1\n"
2  "fxvsubhm %[var], %[var], %[help], 0\n"
3  "fxvsubhm %[help], %[min], %[var], 0\n"
4  "fxvcmph %[help]\n"
5  "fxvsel %[help], %[zero], %[help], 1\n"
6  "fxvaddhm %[help], %[min], %[help], 0\n"
7  "fxvsel %[var], %[var], %[help], 1\n"
```

First of all, according to the VCR, the helper register gets filled with zeros or the bit which was previously added. Afterwards, the helper is subtracted from the sweeped parameter (line 2). Thus, subtracting a zero means keeping the bit and nothing will change for this parameter in the following. In line 3, the variable gets subtracted from the minimum to make it possible to determine whether it is above or below the minimum. Therefore in line 4 the register is compared to zero and the result is stored in the VCR. If the result was above zero, thus indicating the variable

was below the minimum, then the difference is added to the minimum value in line 5 and 6 to prevent crosstalk by setting many parameters to the minimum. Last but not least, the right parameter is chosen at the end.

A.3. Assembly of maximum algorithm

Determining the maximum value of the CADC fast, using the vector unit, has some caveats. That is because each CADC has 8 bit precision and the output is unsigned. The vector unit however only supports signed operations, thus the most significant bit marks a range of 128 LSB to 255 LSB, but vector operations will treat these values as negatives because of the two's complement. Shifting the values one by right would prevent this problem, but in this case 1 bit of precision would be lost.

`max` − `cadc`	`max` $\in \{0, ..., 127\}$	`max` $\in \{128, ..., 255\}$
`cadc` $\in \{0, ..., 127\}$	(I) > 0: `max` bigger < 0: `cadc` bigger	(II) > 0: Never happens < 0: `max` bigger
`cadc` $\in \{128, ..., 255\}$	(III) > 0: `cadc` bigger < 0: Never happens	(IV) > 0: `max` bigger < 0: `cadc` bigger

Table A.1: Different cases if one calculates `max` − `cadc` with an saturating function for unsigned 8-bit integers on 8-bit signed arithmetic.

There are four different cases if the latest obtained value from the CADC is subtracted with a saturating function from the currently determined maximum. These are visualized in table A.1. Normally if the result would be below zero, one would update the maximum because the current CADC read is bigger. But as seen in table A.1, this is not the case for all cases. In case (I) the MSB is not set and the subtraction delivers a result as expected. The same happens for case (IV), for example a CADC read of 255 LSB, corresponding to a two's complement of -1, subtracted from a maximum value of 128 LSB, corresponding to a two's complement of -128, gives a negative value and thus the maximum would be updated. Case (II) shows a different behaviour. The maximum value is treated as negative value, while the latest CADC read is positive. A saturating subtraction always delivers a negative value, so the maximum would be updated. In reality the maximum is bigger because of the MSB, so this step would be wrong. The same happens for case (III), here the subtraction is always positive while the current CADC read is bigger than the maximum. The following assembly code takes care of this problem.

```
1  "fxvshb %[shift], %[cadc], -7\n"
2  "fxvshb %[diff], %[max], -7\n"
3  "fxvsubbm %[shift], %[shift], %[diff]\n"
4  "fxvsubbfs %[diff], %[max], %[cadc]\n"
5  "fxvcmpb %[shift]\n"
6  "fxvaddbm %[diff], %[diff], %[adder], 3\n"
7  "fxvcmpb %[diff]\n"
8  "fxvsel %[max], %[max], %[cadc], 1\n"
```

To overcome the problem one first has to distinguish between the different cases. By shifting the values from `max` and `cadc` for 7 bits to the right, only the MSB is left. This is done in line 1 and 2. Subtracting both resulting values in line 3 gives a measure which case is reached. A result of 0 says that either case (I) or (IV) is given, the comparison is done in line 5. The saturating subtraction of `cadc` from `max` is done in line 4. For case (I) and (IV) a `adder` register is added by a modulo operation in line 6. This register is filled with a number of 128, or -128 in signed arithmetic, respectively, to change the sign and to align all four cases. According to the sign, the new maximum is now selected in line 7 and 8.

A.4. Differential equation for synaptic input

The voltage at the synaptic input with an incoming spike can be described with:

$$V_{\text{syn, exc}}(t) = V_{\text{ref, exc}} - \Delta V_{\text{spk}} \cdot \exp\left(-\frac{t}{\tau_{\text{syn}}}\right).$$

So the current onto the membrane can be calculated with:

$$I_{\text{syn, exc}}(t) = g_{\text{syn}} \cdot \left(V_{\text{ref, exc}} - V_{\text{syn}}(t)\right) = \dot{Q}_{\text{syn, exc}}.$$

The leak current depends on the membrane potential with $g_{\text{leak}} = \frac{C_{\text{mem}}}{\tau_{\text{mem}}}$:

$$I_{\text{leak}}(t) = g_{\text{leak}} \cdot (V_{\text{leak}} - V_{\text{mem}}(t)) = \dot{Q}_{\text{leak}}.$$

With $Q_{\text{mem}}(t) = C_{\text{mem}} \cdot V_{\text{mem}}(t)$ the change of the membrane potential can be calculated with the given currents from synaptic input and leak:

$$\dot{Q}_{\text{mem}} = C_{\text{mem}} \cdot \dot{V}_{\text{mem}} = \dot{Q}_{\text{syn, exc}} + \dot{Q}_{\text{leak}}.$$

This leads to a first order differential equation:

$$\dot{V}_{\text{mem}} = \frac{g_{\text{syn}}}{C_{\text{mem}}} \cdot \Delta V_{\text{spk}} \cdot \exp\left(-\frac{t}{\tau_{\text{syn}}}\right) + \frac{1}{\tau_{\text{mem}}} \cdot V_{\text{leak}} - \frac{1}{\tau_{\text{mem}}} V_{\text{mem}}.$$

Solving the equation gives

$$V_{\text{mem}}(t) = \frac{\Delta V_{\text{spk}} \cdot \tau_{\text{syn}}}{\frac{C_{\text{mem}}}{g_{\text{syn}}} \cdot \left(\frac{\tau_{\text{syn}}}{\tau_{\text{mem}}} - 1\right)} \cdot \exp\left(-\frac{t}{\tau_{\text{syn}}}\right) + V_{\text{leak}} + k_1 \cdot \exp\left(-\frac{t}{\tau_{\text{mem}}}\right),$$

with k_1 as integration constant. The constraint $V_{\text{mem}} = V_{\text{leak}}$ for $t = 0$ delivers the solved equation:

$$\begin{aligned} V_{\text{mem}}(t) = & \frac{\Delta V_{\text{syn}} \cdot \tau_{\text{syn}}}{\frac{C_{\text{mem}}}{g_{\text{syn}}} \cdot \left(1 - \frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)} \cdot \\ & \left(\exp\left(-\frac{t}{\tau_{\text{mem}}}\right) - \exp\left(-\frac{t}{\tau_{\text{syn}}}\right)\right) + V_{\text{leak}}. \end{aligned} \tag{A.1}$$

Neglecting the leak, i.e. $\tau_{\text{mem}} \to \infty$, the membrane potential will rise according to equation 3.6 with the following function of time:

$$V_{\text{mem}}(t) = \frac{\Delta V_{\text{syn}} \cdot \tau_{\text{syn}}}{\frac{C_{\text{mem}}}{g_{\text{syn}}}} \cdot \left(1 - \exp\left(\frac{t}{\tau_{\text{syn}}}\right)\right) + V_{\text{leak}}.$$

The time t_{max}, indicating when the maximum of equation A.1 is reached, can be determined by the derivative $\frac{\partial V_{\text{mem}}}{\partial t} \stackrel{!}{=} 0$ giving

$$t_{max} = \frac{\tau_{\text{syn}} \cdot \tau_{\text{mem}} \cdot \ln\left(\frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)}{\tau_{\text{syn}} - \tau_{\text{mem}}}.$$

Inserted into equation A.1 gives equation 3.8:

$$\begin{aligned}\Delta V_{\text{mem}} = &\frac{\Delta V_{\text{syn}} \cdot \tau_{\text{syn}}}{\frac{C_{\text{mem}}}{g_{\text{syn}}} \cdot \left(1 - \frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)} \cdot \\ &\left(\exp\left(-\frac{\tau_{\text{syn}} \cdot \ln\left(\frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)}{\tau_{\text{syn}} - \tau_{\text{mem}}}\right) - \exp\left(-\frac{\tau_{\text{mem}} \cdot \ln\left(\frac{\tau_{\text{syn}}}{\tau_{\text{mem}}}\right)}{\tau_{\text{syn}} - \tau_{\text{mem}}}\right)\right).\end{aligned}$$

With leak potentially disabled ($\tau_{\text{mem}} \to \infty$), it results in equation 3.6

$$\Delta V_{\text{mem}} = \frac{\Delta V_{\text{syn}} \cdot \tau_{\text{syn}}}{\frac{C_{\text{mem}}}{g_{\text{syn}}}},$$

thus indicating that it is the right analytical solution for the problem.

A.5. Synapse bug

The first idea was to calibrate all neurons at once for weight-rate and rate-rate and use cherry-picked neurons at the end, which match the desired characteristic best. First results looked promising and all neurons matched more or less the desired characteristics, even cherry-picking would not be necessary. With the calibrated values, the experiment however did not work as expected and the output rates were much smaller as expected and observed by the calibration. The problem that occurred can be seen in figure A.1. When for every neuron 16 synapses are active with weight 63, like it is used in the calibration, the amplitude ΔV_{spk} is much bigger compared to only one synapse column with 16 synapses active of the according neuron.

This effect was not expected and makes it impossible to use a spike based calibration for all neurons, while just some of them are used later in the experiment. Because of this, also the problem with the rate based calibration of the excitatory synaptic time constant $\tau_{\text{syn, exc}}$, shown in section 3.5.4, could be explained. It is just not easily possible to control ΔV_{spk} generated by a spike which travelled through the synapse array.

In order to better understand the problem, a sweep was made measuring one neuron receiving input from 16 synapses with weight 63 and determining the resulting

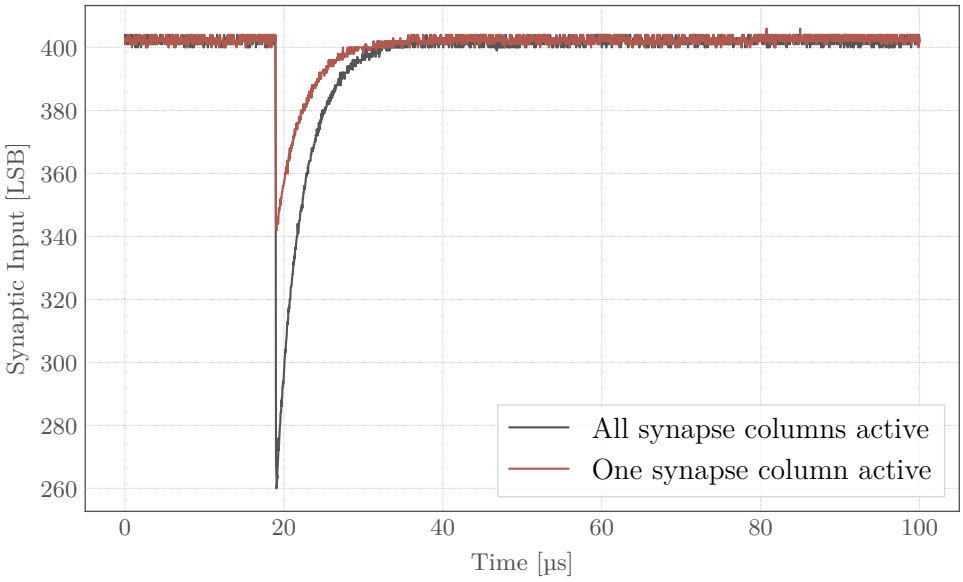

Figure A.1: Incoming spike on the synaptic input. With only the 16 synapses of the target neuron active (weight 63), the amplitude ΔV_{spk} is way smaller (red curve) than for all 16 synapses of all neurons active (black curve). Data measured for neuron 0 on chip 22 on setup 63.

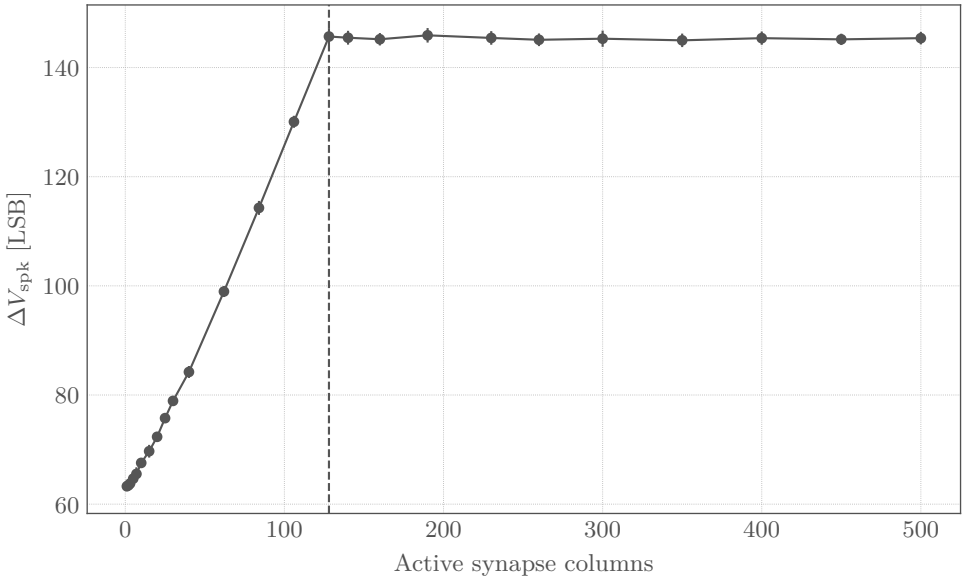

Figure A.2: ΔV_{spk} of neuron 0 with 16 synapses active (weight 63). The amount of active synapses of other neurons was sweeped according to the neuron number. Amplitudes get higher for more synapse columns active until neuron 128 is reached (marked by the dashed line). Thus, indicating it is just a quadrant-wide effect because quadrant 0 ends there. Mean of 50 measurements taken for each data point, on chip 22 on setup 63.

ΔV_{spk} for a different number of other neurons also receiving input from their 16 synapses in the same row. 50 measurements per data point were taken and mean and standard deviation are visible in figure A.2. It is clearly visible that for a small amount of active neuron columns with 16 synapses active, the amplitude of a incoming spike is about 60 LSB. For more neuron columns this rises linearly and reaches amplitudes above 140 LSB for 127 neurons columns active. Afterwards ΔV_{spk} is the same even when more neuron columns are active. This is the case because neuron 128 is already assigned to quadrant 1, thus indicating it is just a quadrant-wide effect. The border of quadrant 0 is marked with the dashed line.

Because of this bug, the calibration has to be done individually for every neuron that ΔV_{spk} is also the same for the experiment with just 18 neurons active. For the supersynapses also a lower synapse current had to be chosen which defines the amplitude of the synaptic current pulses for each quadrant. Thus, 8 neurons are calibrated on quadrant 0 for weight-rate, while 10 neurons on quadrant 1 are calibrated for rate-rate (8 for CPU1 and 2 for the motor neurons). Also by just using the neurons on the same chip hemisphere makes it possible to tune the weights on the same PPU, thus the other PPU is disabled.

The reason for this is expected to result from a transistor included in every synapse instance. Every incoming spike slightly changes the Gate-Source voltage shared across all synapses within a quadrant. With only some synapses spiking simultaneously this effect is just barely visible. But with 128x16 synapses spiking at once, the voltage shift significantly changes the drain current of the transistor resulting in a visible effect. For normal experiments this bug should not be a hard problem, because normally such a big number of synapses do not fire at once. Calibrating all neurons at once with simultaneous spikes in many synapses however makes the problem appear.

References

Guo-qiang Bi and Mu-ming Poo. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, 24(1):139–166, 2001. doi: 10.1146/annurev.neuro.24.1.139. URL https://doi.org/10.1146/annurev.neuro.24.1.139. PMID: 11283308.

M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, 2014. doi: 10.1109/TNNLS.2013.2293637.

Andrew Bogaard, Jack Parent, Michal Zochowski, and Victoria Booth. Interaction of cellular and network mechanisms in spatiotemporal pattern formation in neuronal networks. *Journal of Neuroscience*, 29(6):1677–1687, 2009. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.5218-08.2009. URL https://www.jneurosci.org/content/29/6/1677.

Mikhail Borodin, Kaushik De, Jose Garcia Navarro, Dmitry Golubkov, Alexei Klimentov, Tadashi Maeno, David South, and Alexandre Vaniachine. Big data processing in the atlas experiment: Use cases and experience. *Procedia Computer Science*, 66:609–618, 2015. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.11.069. URL https://www.sciencedirect.com/science/article/pii/S1877050915034183. 4th International Young Scientist Conference on Computational Science.

Romain Brette and Wulfram Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637–3642, 2005. doi: 10.1152/jn.00686.2005. URL https://doi.org/10.1152/jn.00686.2005. PMID: 16014787.

C.G. Butler, D.J.C. Fletcher, and Doreen Watler. Nest-entrance marking with pheromones by the honeybee-apis mellifera l., and by a wasp, vespula vulgarjs l. *Animal Behaviour*, 17:142–147, 1969. ISSN 0003-3472. doi: https://doi.org/10.1016/0003-3472(69)90122-5. URL https://www.sciencedirect.com/science/article/pii/0003347269901225.

Stefano Cavallari, Stefano Panzeri, and Alberto Mazzoni. Comparison of the dynamics of neural interactions between current-based and conductance-based integrate-and-fire recurrent networks. *Frontiers in neural circuits*, 8:12–12, Mar 2014. ISSN 1662-5110. doi: 10.3389/fncir.2014.00012. URL https://pubmed.ncbi.nlm.nih.gov/24634645. 24634645[pmid].

Lars Chittka and Adrian Dyer. Your face looks familiar. *Nature*, 481(7380):154–155, Jan 2012. ISSN 1476-4687. doi: 10.1038/481154a. URL https://doi.org/10.1038/481154a.

Marie Dacke, Marcus J. Byrne, Clarke H. Scholtz, and Eric J. Warrant. Lunar orientation in a beetle. *Proceedings. Biological sciences*, 271(1537):361–365, Feb 2004. ISSN 0962-8452. doi: 10.1098/rspb.2003.2594. URL https://pubmed.ncbi.nlm.nih.gov/15101694. 15101694[pmid].

Philipp Dauer. Characterization of silicon neurons on hicann-x v2. Bachelorarbeit, Universität Heidelberg, 2020.

M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.

M. V. DeBole, B. Taba, A. Amir, F. Akopyan, A. Andreopoulos, W. P. Risk, J. Kusnitz, C. Ortega Otero, T. K. Nayak, R. Appuswamy, P. J. Carlson, A. S. Cassidy, P. Datta, S. K. Esser, G. J. Garreau, K. L. Holland, S. Lekuch, M. Mastro, J. McKinstry, C. di Nolfo, B. Paulovicks, J. Sawada, K. Schleupen, B. G. Shaw, J. L. Klamo, M. D. Flickner, J. V. Arthur, and D. S. Modha. Truenorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5):20–29, 2019. doi: 10.1109/MC.2019.2903009.

Carlos Eyzaguirre and Stephen W. Kuffler. Further study of soma, dendrite, and axon excitation in single neurons. *The Journal of General Physiology*, 39(1): 121–153, 1955. ISSN 0022-1295. doi: 10.1085/jgp.39.1.121. URL `http://jgp.rupress.org/content/39/1/121`.

Diasynou Fioravante and Wade G Regehr. Short-term forms of presynaptic plasticity. *Current Opinion in Neurobiology*, 21(2):269–274, 2011. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2011.02.003. URL `https://www.sciencedirect.com/science/article/pii/S0959438811000298`. Synaptic function and regulation.

S. Friedmann, J. Schemmel, A. Grübl, A. Hartel, M. Hock, and K. Meier. Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE Transactions on Biomedical Circuits and Systems*, 11(1):128–142, Feb 2017. ISSN 1940-9990. doi: 10.1109/TBCAS.2016.2579164.

Simon Friedmann and Christian Pehle. Nux user guide, 09 2020. URL `https://flagship.kip.uni-heidelberg.de/jss/FileExchange/Guidebook_NUX.pdf?fID=4363&s=qqdXDg6HuX3&uID=65`.

S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014. doi: 10.1109/JPROC.2014.2304638.

Stanley Heinze and Uwe Homberg. Maplike representation of celestial e-vector orientations in the brain of an insect. *e-Neuroforum*, 13(2):62–63, 2007. doi: doi:10.1515/nf-2007-0205. URL `https://doi.org/10.1515/nf-2007-0205`.

Stanley Heinze and Steven M. Reppert. Sun compass integration of skylight cues in migratory monarch butterflies. *Neuron*, 69(2):345–358, 2011. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2010.12.025. URL `https://www.sciencedirect.com/science/article/pii/S0896627310010731`.

Stanley Heinze, Ajay Narendra, and Allen Cheung. Principles of insect path integration. *Current Biology*, 28(17):R1043–R1058, 2018. ISSN 0960-9822. doi:

https://doi.org/10.1016/j.cub.2018.04.058. URL https://www.sciencedirect.com/science/article/pii/S096098221830530X.

Suzana Herculano-Houzel. The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. *Proceedings of the National Academy of Sciences*, 109(Supplement 1):10661–10668, 2012. ISSN 0027-8424. doi: 10.1073/pnas.1201895109. URL https://www.pnas.org/content/109/Supplement_1/10661.

M. Hock, A. Hartel, J. Schemmel, and K. Meier. An analog dynamic memory array for neuromorphic hardware. In *2013 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4, 2013. doi: 10.1109/ECCTD.2013.6662229.

Matthias Hock. *Modern semiconductor technologies for neuromorphic hardware*. PhD thesis, Universität Heidelberg, July 2014.

Scarlett R. Howard, Aurore Avarguès-Weber, Jair E. Garcia, Andrew D. Greentree, and Adrian G. Dyer. Numerical cognition in honeybees enables addition and subtraction. *Science Advances*, 5(2), 2019. doi: 10.1126/sciadv.aav0961. URL https://advances.sciencemag.org/content/5/2/eaav0961.

Y. Kodama, T. Odajima, E. Arima, and M. Sato. Evaluation of power management control on the supercomputer fugaku. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 484–493, 2020. doi: 10.1109/CLUSTER49012.2020.00069.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL https://doi.org/10.1145/3065386.

Florent Le Moël, Thomas Stone, Mathieu Lihoreau, Antoine Wystrach, and Barbara Webb. The central complex as a potential substrate for vector based navigation. *Frontiers in Psychology*, 10:690, 2019. ISSN 1664-1078. doi: 10.3389/fpsyg.2019.00690. URL https://www.frontiersin.org/article/10.3389/fpsyg.2019.00690.

Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi: 10.1038/nature14539.

Aron Leibfried. On-chip calibration of analog neuromorphic circuits. Bachelorarbeit, Universitat Heidelberg, 2018.

Olli J. Loukola, Clint J. Perry, Louie Coscos, and Lars Chittka. Bumblebees show cognitive flexibility by improving on an observed complex behavior. *Science*, 355(6327):833–836, 2017. ISSN 0036-8075. doi: 10.1126/science.aag2360. URL https://science.sciencemag.org/content/355/6327/833.

Xiaoya Ma, Xianguang Hou, Gregory D. Edgecombe, and Nicholas J. Strausfeld. Complex brain and optic lobes in an early cambrian arthropod. *Nature*, 490 (7419):258–261, Oct 2012. ISSN 1476-4687. doi: 10.1038/nature11495. URL https://doi.org/10.1038/nature11495.

Christian Mayr, Sebastian Höppner, and Steve B. Furber. Spinnaker 2: A 10 million core processor system for brain simulation and machine learning. *CoRR*, abs/1911.02385, 2019. URL http://arxiv.org/abs/1911.02385.

A. Mehmood, M. J. Iqbal, H. Dawood, and P. Guo. Computationally efficient low power neuron model for digital brain. In *2018 14th International Conference on Computational Intelligence and Security (CIS)*, pages 1–5, 2018. doi: 10.1109/CIS2018.2018.00009.

Randolf Menzel and Martin Giurfa. Cognitive architecture of a mini-brain: the honeybee. *Trends in Cognitive Sciences*, 5(2):62–71, 2001. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(00)01601-6. URL https://www.sciencedirect.com/science/article/pii/S1364661300016016.

Eric Müller, Sebastian Schmitt, Christian Mauch, Sebastian Billaudelle, Andreas Grübl, Maurice Güttler, Dan Husmann, Joscha Ilmberger, Sebastian Jeltsch, Jakob Kaiser, Johann Klähn, Mitja Kleider, Christoph Koke, José Montes, Paul Müller, Johannes Partzsch, Felix Passenberg, Hartmut Schmidt, Bernhard Vogginger, Jonas Weidner, Christian Mayr, and Johannes Schemmel. The operating system of the neuromorphic brainscales-1 system, 2020.

Richard Naud, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner. Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 99(4):335, Nov 2008. ISSN 1432-0770. doi: 10.1007/s00422-008-0264-7. URL https://doi.org/10.1007/s00422-008-0264-7.

Bente Pakkenberg, Dorte Pelvig, Lisbeth Marner, Mads J. Bundgaard, Hans Jørgen G. Gundersen, Jens R. Nyengaard, and Lisbeth Regeur. Aging and the human neocortex. *Experimental Gerontology*, 38(1):95–99, 2003. ISSN 0531-5565. doi: https://doi.org/10.1016/S0531-5565(02)00151-1. URL https://www.sciencedirect.com/science/article/pii/S0531556502001511. Proceedings of the 6th International Symposium on the Neurobiology and Neuroendocrinology of Aging.

Ferdinand Peper. The end of moore's law: Opportunities for natural computing? *New Generation Computing*, 35(3):253–269, Jul 2017. ISSN 1882-7055. doi: 10.1007/s00354-017-0020-4. URL https://doi.org/10.1007/s00354-017-0020-4.

A. V. Popov, A. I. Peresleni, P. V. Ozerskii, E. E. Shchekanov, and E. V. Savvateeva-Popova. On the role of the protocerebral bridge in the central complex of drosophila melanogaster brain in control of courtship behavior and sound production. *Journal of Evolutionary Biochemistry and Physiology*, 39(6):655–666, Nov 2003. ISSN 1608-3202. doi: 10.1023/B:JOEY.0000023486.75353.f8. URL https://doi.org/10.1023/B:JOEY.0000023486.75353.f8.

Q. Rao and J. Frtunikj. Deep learning for self-driving cars: Chances and challenges. In *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*, pages 35–38, 2018.

A.J. Riveros and R.B. Srygley. Do leafcutter ants, atta colombica, orient their path-integrated home vector with a magnetic compass? *Animal Behaviour*, 75

(4):1273–1281, 2008. ISSN 0003-3472. doi: https://doi.org/10.1016/j.anbehav.2007.09.030. URL https://www.sciencedirect.com/science/article/pii/S0003347208000201.

Robert H. Rogne, Torleiv H. Bryne, Thor I. Fossen, and Tor A. Johansen. Mems-based inertial navigation on dynamically positioned ships: Dead reckoning. *IFAC-PapersOnLine*, 49(23):139–146, 2016. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2016.10.334. URL https://www.sciencedirect.com/science/article/pii/S2405896316319218. 10th IFAC Conference on Control Applications in Marine SystemsCAMS 2016.

Bernhard Ronacher. Path integration in a three-dimensional world: the case of desert ants. *Journal of Comparative Physiology A*, 206(3):379–387, May 2020. ISSN 1432-1351. doi: 10.1007/s00359-020-01401-1. URL https://doi.org/10.1007/s00359-020-01401-1.

Johannes Schemmel, Sebastian Billaudelle, Phillip Dauer, and Johannes Weis. Accelerated analog neuromorphic computing. *arXiv preprint arXiv:2003.11996*, 2020.

Dominik Schmidt. Automated characterization of a wafer-scale neuromorphic hardware system. Masterarbeit, Universität Heidelberg, 2014.

Korbinian Schreiber. *Accelerated neuromorphic cybernetics*. PhD thesis, Universität Heidelberg, January 2021.

Sebastian Schwarz, Laurence Albert, Antoine Wystrach, and Ken Cheng. Ocelli contribute to the encoding of celestial compass information in the australian desert ant melophorus bagoti. *Journal of Experimental Biology*, 214(6):901–906, 2011. ISSN 0022-0949. doi: 10.1242/jeb.049262. URL https://jeb.biologists.org/content/214/6/901.

Hayk Shoukourian, Torsten Wilde, Axel Auweter, and Arndt Bode. Predicting the energy and power consumption of strong and weak scaling hpc applications. *Supercomputing Frontiers and Innovations*, 1(2), 2014. ISSN 2313-8734. URL https://superfri.org/superfri/article/view/9.

S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, and S. Jain. Machine translation using deep learning: An overview. In *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pages 162–167, 2017. doi: 10.1109/COMPTELIX.2017.8003957.

Mandyan V. Srinivasan, Shaowu Zhang, Monika Altwein, and Jürgen Tautz. Honeybee navigation: Nature and calibration of the "odometer". *Science*, 287(5454):851–853, 2000. ISSN 0036-8075. doi: 10.1126/science.287.5454.851. URL https://science.sciencemag.org/content/287/5454/851.

Thomas Stone, Barbara Webb, Andrea Adden, Nicolai Ben Weddig, Anna Honkanen, Rachel Templin, William Wcislo, Luca Scimeca, Eric Warrant, and Stanley Heinze. An anatomically constrained model for path integration in the bee brain. *Current Biology*, 27(20):3069–3085.e11, 2017. ISSN 0960-9822. doi: https://doi.org/10.1016/j.cub.2017.08.052. URL https://www.sciencedirect.com/science/article/pii/S0960982217310904.

Piergiorgio Strata and Robin Harvey. Dale's principle. *Brain Res. Bull.*, 50:349—50, 01 1999.

Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, Mar 2019. ISSN 0893-6080. doi: 10.1016/j.neunet.2018.12.002. URL http://dx.doi.org/10.1016/j.neunet.2018.12.002.

Misha V. Tsodyks and Henry Markram. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proceedings of the National Academy of Sciences*, 94(2):719–723, 1997. ISSN 0027-8424. doi: 10.1073/pnas.94.2.719. URL https://www.pnas.org/content/94/2/719.

Mark Vogelsberger, Federico Marinacci, Paul Torrey, and Ewald Puchwein. Cosmological simulations of galaxy formation, 2019.

R Wehner. Astronavigation in insects. *Annual Review of Entomology*, 29(1):277–298, 1984. doi: 10.1146/annurev.en.29.010184.001425. URL https://doi.org/10.1146/annurev.en.29.010184.001425.

Johannes Weis. Inference with artificial neural networks on neuromorphic hardware. Master's thesis, Universität Heidelberg, 9 2020.

Johannes Weis, Philipp Spilger, Sebastian Billaudelle, Yannik Stradmann, Arne Emmel, Eric Müller, Oliver Breitwieser, Andreas Grübl, Joscha Ilmberger, Vitali Karasenko, Mitja Kleider, Christian Mauch, Korbinian Schreiber, and Johannes Schemmel. Inference with artificial neural networks on analog neuromorphic hardware, 2020.

Matthias Wittlinger, Rüdiger Wehner, and Harald Wolf. The ant odometer: Stepping on stilts and stumps. *Science*, 312(5782):1965–1967, 2006. ISSN 0036-8075. doi: 10.1126/science.1126912. URL https://science.sciencemag.org/content/312/5782/1965.

Jasmine M. Yakubowski, Glyn A. McMillan, and John R. Gray. Background visual motion affects responses of an insect motion-sensitive neuron to objects deviating from a collision course. *Physiological Reports*, 4(10):e12801, 2016. doi: https://doi.org/10.14814/phy2.12801. URL https://physoc.onlinelibrary.wiley.com/doi/abs/10.14814/phy2.12801.

Zhihao Zheng, J. Scott Lauritzen, Eric Perlman, Camenzind G. Robinson, Matthew Nichols, Daniel Milkie, Omar Torrens, John Price, Corey B. Fisher, Nadiya Sharifi, Steven A. Calle-Schuler, Lucia Kmecova, Iqbal J. Ali, Bill Karsh, Eric T. Trautman, John A. Bogovic, Philipp Hanslovsky, Gregory S.X.E. Jefferis, Michael Kazhdan, Khaled Khairy, Stephan Saalfeld, Richard D. Fetter, and Davi D. Bock. A complete electron microscopy volume of the brain of adult drosophila melanogaster. *Cell*, 174(3):730–743.e22, 2018. ISSN 0092-8674. doi: https://doi.org/10.1016/j.cell.2018.06.019. URL https://www.sciencedirect.com/science/article/pii/S0092867418307876.

Acknowledgements

I would like to thank

Dr. Johannes Schemmel for his support and the opportunity to carry out this master thesis.

Professor Dr. Peter Fischer for examining this thesis as a second reviewer.

Sebastian Billaudelle for his competent supervision of my work and many helpful tips during the course of my studies.

Korbinian Schreiber for showing me his implementation of the insect experiment and the according explanations.

Johannes Weis, Eric Müller and Philipp Spilger for answering my questions regarding hardware or software.

Jakob Kaiser, Yannik Stradmann, Johannes Weis and Sebastian Billaudelle for proofreading this thesis and the helpful feedback.

Benjamin Cramer, Sebastian Billaudelle, Philipp Dauer and Jakob Kaiser for the entertaining time at the KIP.

The whole Electronic Vision(s) group for all the combined achievements and groundwork that enabled this thesis.

Simon Rosenkranz for the entertaining and informative first two master semesters.

My friends and my family for supporting me during the course of my studies.

The work carried out in this Master Thesis used systems, which received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements Nos. 720270, 785907 and 945539 (Human Brain Project, HBP).

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, April 01, 2021 .....
(signature)