Denis Alevi

Investigation and Comparison between Emulations of a Simple Chain Network on a HICANNv2 Wafer Module and Simulations with NEST

Internship Report
January 2015

Abstract

The BrainScaleS Project and a subproject of the Human Brain Project aim to better understand the human brains communication and information processing structure and to develop Neuromorphic Hardware inspired by the brains architecture. Such Neuromorphic Hardware makes it possible to physically emulate neuroscientific models on different time and spatial scales where todays computers face limitations of time and energy consumption due to the complexity of most biologically relevant models. The scope of this internship was to learn the tools and techniques used to simulate neuroscientific model networks using the PyNN environment and to compare results from emulating the networks on neuromorphic hardware with results on simulators such as NEST or NEURON. A simple neuron chain was simulated with NEST and emulated on a HICANNv2 wafer scale system. It was possible to reproduce similar spiking patterns.
While working on the internship, interactive widgets with IPython where programmed. These help to better understand the influence of different cell or network parameters on membrane potential or spiking patterns as the parameters can be changed interactively while directly observing the effect on the output.

Contents

1 Introduction

1.1 The High Input Count Analog Neural Network (HICANN) chip

In this internship physical emulation experiments have been carried out on the HICANNv2 wafer module.
The following is a short introduction on building structure and components of the hardware used in this internship. For detailed technical information see *HBP SP9 partners* (2014).

1.1.1 Analog neuron and synapse circuits

One HICANN wafer module contains 384 HICANN microchips. Each one of these chips can implement up to 512 neurons and 114688 synapses (*Jeltsch*, 2014). Every neuron has two synaptic input circuits, which reversal potentials can be set individually. Those two synaptic inputs are connected to 224 synapses. Neighbouring neuron circuits within blocks of 64 can be combined to form one larger neuron with more connected synapses. Therefore one block of 64 combined neuron circuits can implement a neuron with $64 \times 224 = 14336$ individual inputs.
The dynamics of the analog neuron circuits have a configurable speedup of 10^3 to 10^5 compared to biological time. That means a speedup of 10^4 results in a emulation time of 10^{-4} seconds for one second in biological time.

1.1.2 Implemented Neuron Model

The neuron circuits implement the Adaptive Exponential Integrate-and-Fire (AdEx) model (*Gerstner et al.*, 2014), which is characterized by two differential equations:

$$\tau_{\mathrm{m}} \frac{\mathrm{d}V}{\mathrm{d}t} = -(V - V_{\mathrm{rest}}) + \Delta_T \exp\left(\frac{V - V_{\mathrm{thresh}}}{\Delta_T}\right) - \frac{\tau_{\mathrm{m}}}{C_{\mathrm{m}}} \cdot w + \frac{\tau_{\mathrm{m}}}{C_{\mathrm{m}}} \cdot I \tag{1.1}$$

$$\tau_w \frac{\mathrm{d}w}{\mathrm{d}t} = a(V - V_{\mathrm{rest}}) - w \tag{1.2}$$

The equations are expressed in terms of the parameters used in PyNN, where V is the membrane potential of the neuron, τ_{m} the membrane time constant, V_{rest} the resting membrane potential (also called leak reversal potential), Δ_T the slope factor

controlling the non-linear dynamics of the exponential term, C_m the membrane capacitance, I the injection current, w the adaptation variable, a the subthreshold adaptation conductance and τ_w the adaptation time constant.
Whenever the membrane potential V reaches a spike detection threshold V_spike, a spike is detected and the membrane potential is reset to the value V_reset. At the same time the adaptation variable w is increased by the spike-triggered adaptation constant b. This can be expressed by following equations:

$$\text{if} \quad V(t) = V_\mathrm{spike} \begin{cases} \lim\limits_{\delta \to 0; \delta > 0} V(t+\delta) = V_\mathrm{reset} \\ \lim\limits_{\delta \to 0; \delta > 0} w(t+\delta) = w + b \end{cases} \tag{1.3}$$

Therefore for positive b the excitability is reduced after each spike and is increasing again to its default excitability following equation 1.2. Additionally after a spike the neuron cannot be excited at all for a refractory time τ_refrac.
The exponential term of equation 1.1 and the adaptation w can be turned off on the hardware, making it possible to emulate different Leaky Integrate and Fire (LIF) models.

1.2 The PyNN Interface

Neural network models in this internship were written using the PyNN interface. PyNN is a description language for neuron networks implemented in Python. Code written in PyNN can be executed with different simulation backends such as Nest, Neuron or Brian. To use the same description language for emulations on the HICANN wafer system, the PyNN compatible software interface PyHMF was developed. Through several software layers mapping of simulated neurons to hardware neurons is done and calibration parameters for the hardware are used to achieve better uniformity between different hardware neurons. For more details see *HBP SP9 partners* (2014).

2 Tools

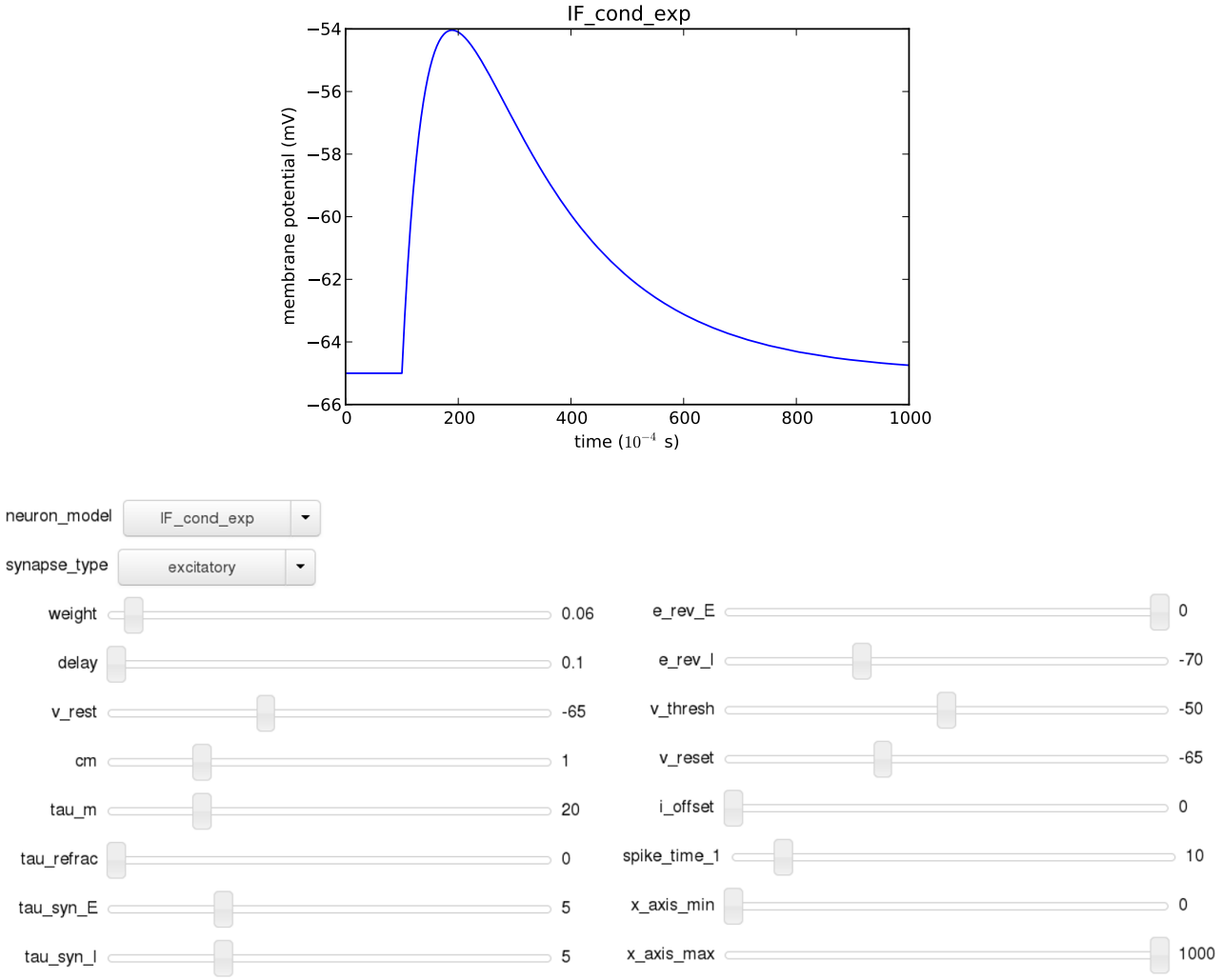


Figure 2.1: IPython notebook with interactive widgets for analyzing neuron model behaviour for different neuron models and parameter settings. Units of time and time constant parameters are ms, unit of c_{m} is nF, unit of *weight* is nA and unit of voltages and potentials is mV. The plot result changes interactively when moving the sliders.

To gain a better understanding of the dependency of different models and networks on certain parameters, interactive Widgets where written with IPython Notebook to make it possible to change parameters and see the effect on model characteristics (e.g. membrane potential) directly without having to rerun the simulation manually for each parameter set. This is useful only for small networks as the simulation is still rerun automatically each time a parameter changes. In figure 2.1 an interactive widget simulating a LIF neuron with fixed threshold and decaying post synaptic conductance (IF_cond models in PyNN) is shown. The parameters can be changed by sliding the sliders and for a set of parameters the neuron model can be switched between the IF_cond_exp and IF_cond_alpha, simulating an exponential and alpha function shaped post-synaptic conductance, respectively.
Similar interactive simulations were written for other neuron models used in PyNN.

3 Methods and Results

3.1 Chain network model

A simple network model was used to get used to working with PyNN and Python and to reproduce results of previous network emulations on the HICANNv2 wafer scale system. The network used was a feed forward chain consisting of several populations of LIF neurons with fixed threshold and exponentially-decaying post-synaptic conductance (IF_cond_exp neuron model in PyNN). A fixed number of neurons of each population is connected excitatory to each neuron of the following population. The neurons of the first population are excited by a single spike source. An exemplary network with 3 populations, 4 neurons per population and 2 connections between successive populations is shown in figure 3.1.

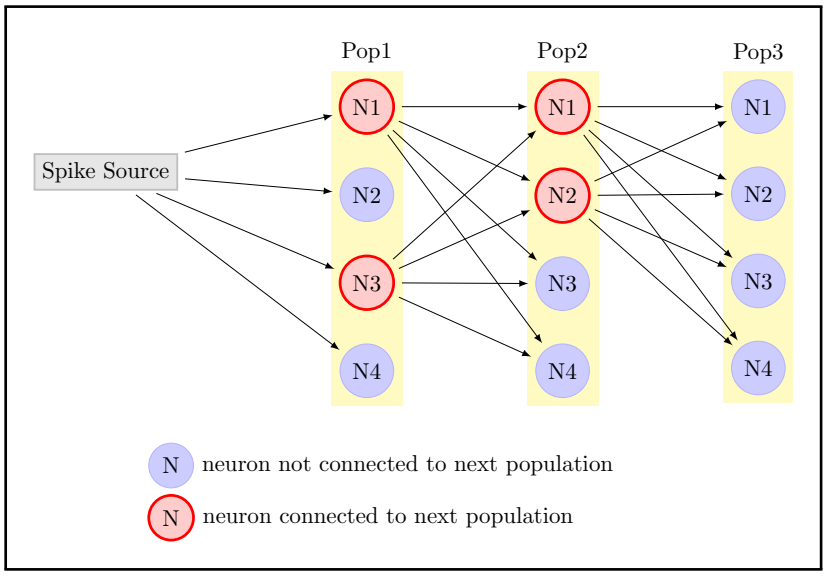


Figure 3.1: Chain Network with 3 Populations of 4 Neurons each and a connectivity of 2. Each neuron of the first population receives excitatory synaptic input from a spike source, each neuron of the following populations receives excitatory synaptic input from 2 random neurons of the preceding population. Arrows indicate excitatory connections.

3.2 Simulations in NEST

First the network described in section 3.1 was simulated using the NEST simulator with the PyNN interface. As in simulation in the network each neuron of the same population receives exactly the same input from the previous population, the simulation can be simplified to single neurons being connected several times depending on the number of connections between neurons of following populations in the original network. This simplification made it possible to create another interactive IPython Notebook (see figure 3.2) to analyze the change of spiking rates of the different populations for different network and cell parameter settings.

For the final simulation a network size of 14 populations with 12 neurons per population and a connectivity of 4 was chosen. To get the same simulation input for each neuron in the network, the spike source is connected 4 times with each neuron of the first population. The simulation parameters were chosen similar to the ones used for the hardware emulations in section 3.3 and the network was simulated for different threshold voltages V_{thresh}. The other parameters stayed unchanged for each simulation and can be found in the appendix A. In figure 3.3 the simulation results are shown. To stimulate the network one spike from the spike source is sent at $t = 10\,\text{ms}$.
For $-48\,\text{mV} \leq V_{\text{thresh}} \leq -45\,\text{mV}$ the simulation results in the same spike counts as can be seen in figure 3.3a. In the figures 3.3e and 3.3f the corresponding membrane potentials for the first population neurons are shown for $V_{\text{thresh}} = -45\,\text{mV}$ and $V_{\text{thresh}} = -48\,\text{mV}$, respectively. We can observe that for lower threshold potentials the membrane potential after the first spike comes closer to the threshold potential, just as would be expected. Due to the equal connectivity between spike source and populations all neurons receive the exact same input in all populations if the first population only spikes once. Therefore we receive the given spike pattern and every neuron will show the exact same voltage trace as the first population.
For $V_{\text{thresh}} = -49\,\text{mV}$ the membrane potential of the first population neurons now reach the threshold a second time after the first spike as can be seen in figure 3.3g. The neurons of the second population now receive two input spikes of which each would trigger two spikes again following a voltage trace similar to the one of neurons in the first population if the neuron of the second population was at rest when the second input spike arrives. If the second spike arrives earlier, the membrane voltage of the receiving neuron would still be below the resting potential which could result in less spiking for the second input as happens e.g. in figure 3.3b where the second population (index 1) spikes only 3 times and not 2×2 as would have happened, if the neuron already reached its resting potential before the second input spike. The spiking still increases from first to second population and that effect even increases for later populations receiving exponentially more input spikes.
For $V_{\text{thresh}} = -50\,\text{mV}$ the second population spikes 4 times and the effect increases even more resulting in even more spikes for populations coming later in the chain.
When the threshold potential is set below the resting potential $V_{\text{thresh}} < V_{\text{rest}}$, the neurons start spiking continuously even without receiving any input as they cross V_{thresh} every time they try to reach V_{rest} as can be seen in figure 3.3d. The resting potential was set to $V_{\text{rest}} = -50\,\text{mV}$ for each simulation.

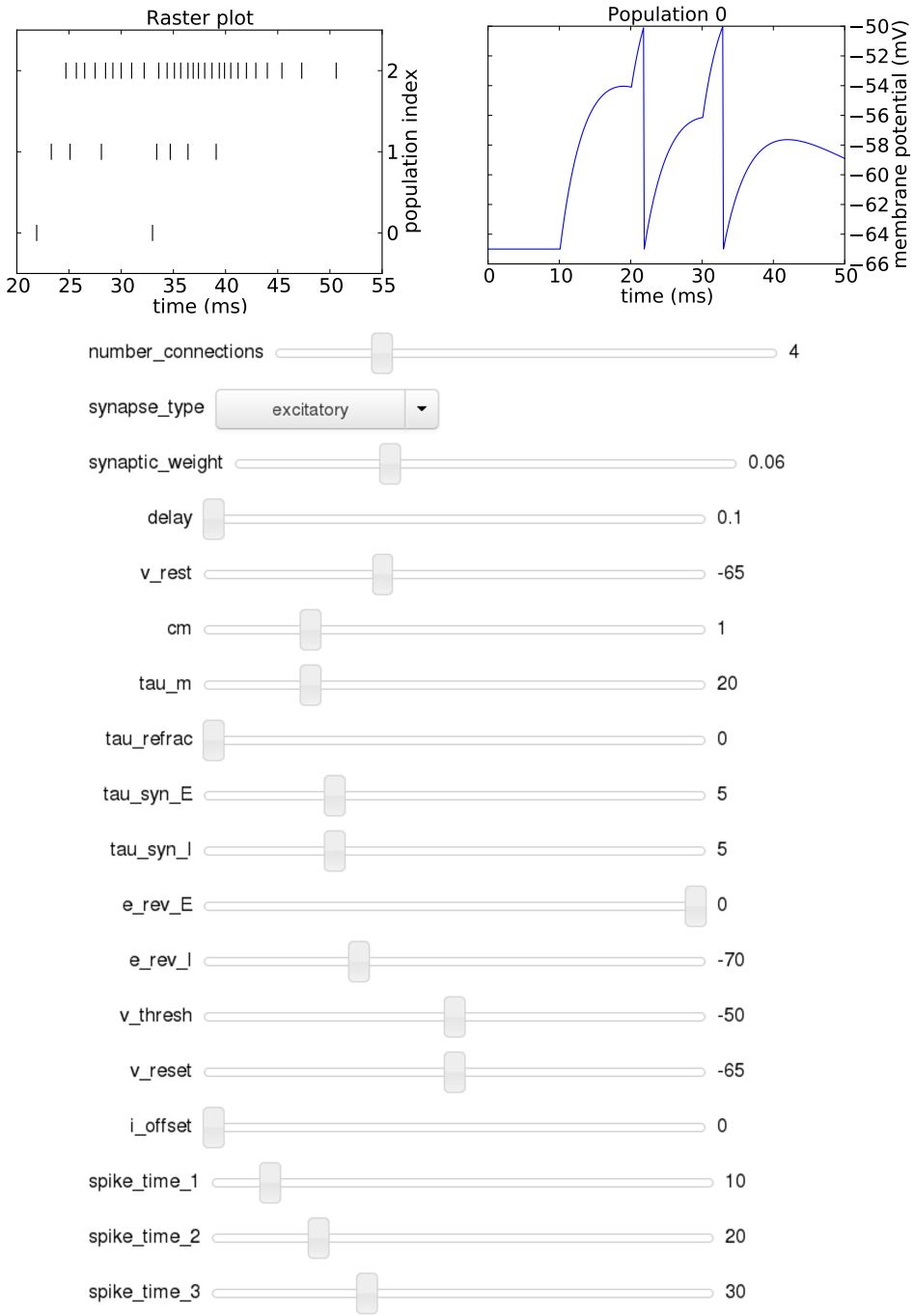
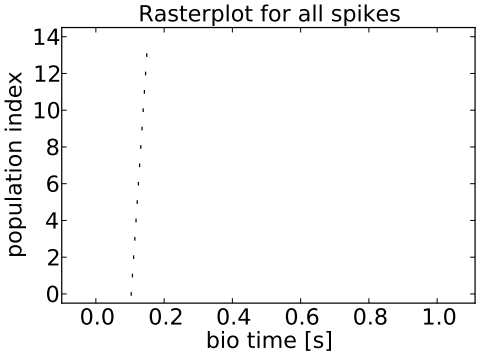
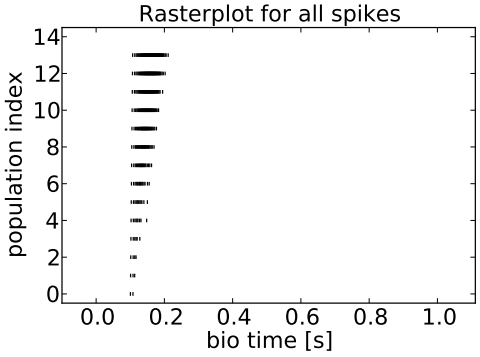


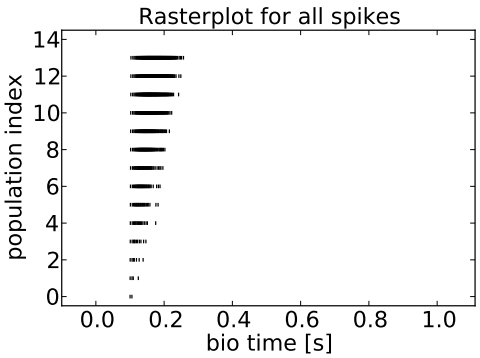
Figure 3.2: IPython notebook with interactive widgets for analyzing the spike behaviour for different parameter settings. A network with 3 populations and 1 neuron per population is simulated. Units of time and time constant parameters are ms, unit of c_{m} is nF, unit of *weight* is nA and unit of voltages as potentials is mV. The resulting raster plot and the membrane voltage of population 0 change interactively when moving the sliders.



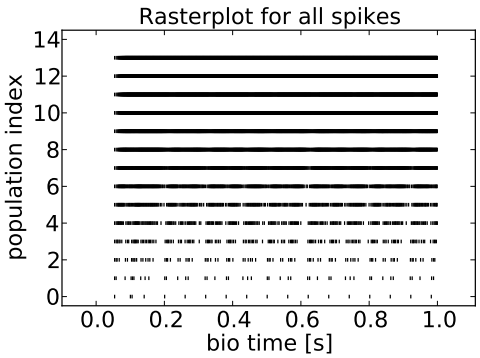
(a) V_{thresh} = -45mV to -48 mV



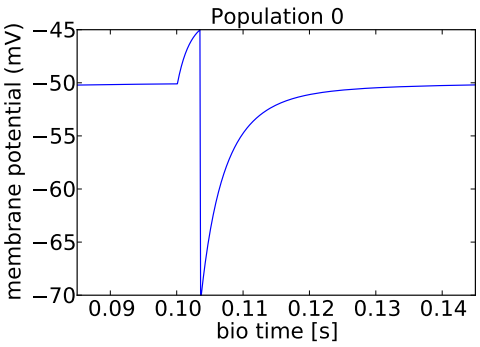
(b) V_{thresh} = -49mV



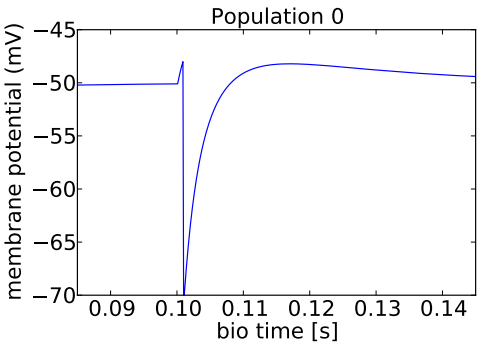
(c) V_{thresh} = -50mV



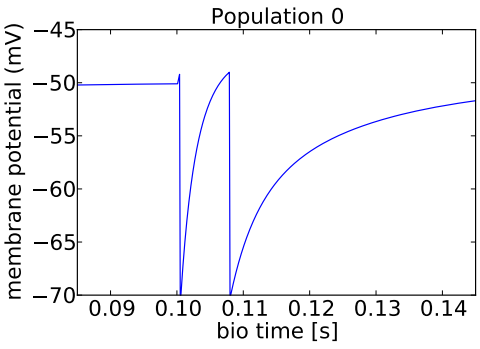
(d) V_{thresh} = -51mV



(e) V_{thresh} = -45mV



(f) V_{thresh} = -48mV



(g) V_{thresh} = -49mV

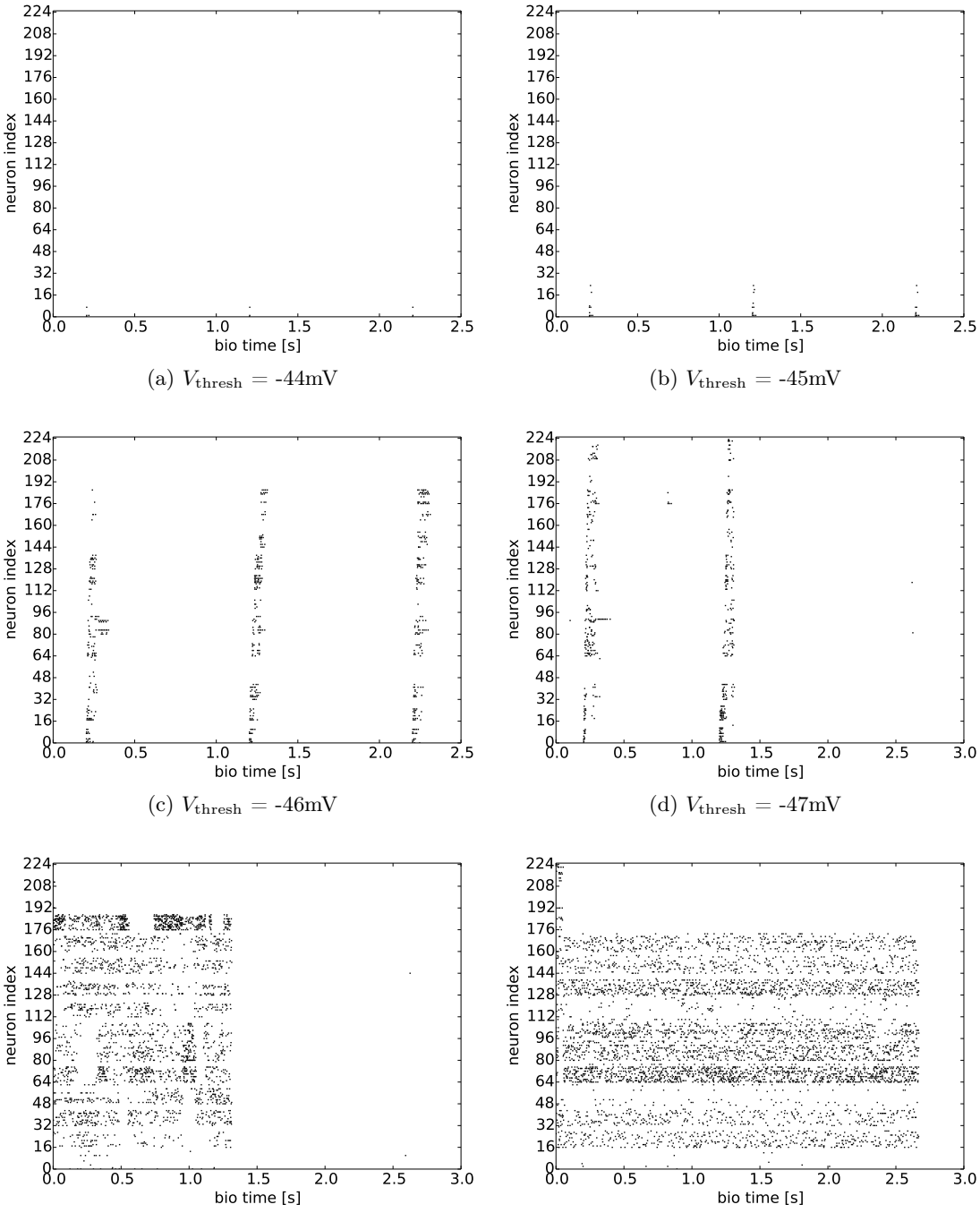
Figure 3.3: Rasterplots ((a)-(d)) and membrane voltage traces for neurons of the first population ((e)-(g)) are shown for simulations with NEST simulator as backend. For each population only one neuron was simulated as each neuron of one population receives the same input.

3.3 Emulation on HICANNv2 wafer

To emulate the chain network on the HICANNv2 wafer, hardware specific code had to be added to control the emulation setup. For the hardware emulation a network with 14 populations of 12 neurons per population and synaptic input for each neuron of a population from 4 neurons of the previous population was implemented. The first population is connected 4 times with the spike source and the spike source sends one spike at the times $t_1 = 1\,\text{ms}$, $t_2 = 1000\,\text{ms}$ and $t_3 = 2000\,\text{ms}$.
For each network neuron four hardware neuron circuits are connected as this is the neuron size, for which the mapping software has been tested before and which most previous hardware experiments used so far.
Therefore one population uses $4 \times 12 = 48$ hardware neurons. Each HICANN consists of 8 output buffers for blocks of 64 hardware neurons each. As the input of each output buffer is limited, only one population per output buffer is emulated. The mapping of simulated neurons to hardware neurons is done neuron by neuron, e.g. if in the simulation code the last neuron of one population is defined just before the first neuron of the next population, they will be implemented on successive hardware neurons. Therefore, to use only one output buffer for one population, $\frac{64-48}{4} = 4$ dummy neurons (each consisting of again 4 hardware neuron circuits) were created in the simulation code to fill up the buffers. Additionally the last output buffer is reserved for background event generation and external spike input. Therefore 7 populations can be emulated on one HICANN chip, of which two are used for this setup to emulate 14 populations. Code written by Sebastian Schmitt and used in the HBP Platform Demonstrator for the HBP Summit 2014 [1] was reused for this purpose. Parameters to receive reasonable spiking patterns were already given in the code and can be found in the appendix B.
To investigate the effect of changing certain parameters, the network was emulated for different values of the threshold voltage V_{thresh} as can be seen in figure 3.4. The figures show exemplary raster plots of all neurons emulated, including the dummy neurons used to fill up the buffers. Therefore between two populations 4 neuron IDs belong to not connected neurons which should not spike at all. This should leave a blank spot between successive populations. The ticks for the neuron index axis where chosen in a way, that they always mark the first neuron of a new population.

The threshold voltage was varied in a range of $-49\,\text{mV} \leq V_{\text{thresh}} \leq -44\,\text{mV}$ as for higher voltages no spikes seem to occur at all and lower voltages result in continuous spiking throughout all chain neurons as can be observed in figure 3.4f.
For $-43\,\text{mV} \leq V_{\text{thresh}}$, the emulation exits with an error. This behaviour seems to be the result of the PyHMF command that reads the array containing times and neuron indices of recorded spikes in an assembly of populations (PyHMF.assembly.getSpikes()). The error occurs in simulation if the spikes array is empty. We can therefore assume, that we don't have any spiking behaviour for the given voltage.

[1] `https://github.com/electronicvisions/hbp_platform_demo`



(a) V_{thresh} = -44mV

(b) V_{thresh} = -45mV

(c) V_{thresh} = -46mV

(d) V_{thresh} = -47mV

(e) V_{thresh} = -48mv

(f) V_{thresh} = -49mV

Figure 3.4: Raster plots for hardware emulation of the chain network on a HICANNv2 wafer module for different values of the threshold voltage V_{thresh} of the emulated neurons.

For $V_{\text{thresh}} = -44\,\text{mV}$ only a few spikes in the first population can be observed. The emulation was repeated 5 times and in none of them did the spiking reach the second population. For $V_{\text{thresh}} = -45\,\text{mV}$ we see that only the first population and a few neurons of the second population start to spike. The emulation was repeated 5 times and for some of the stimulations the spiking reached further then the second population but for most it didn't.
For high voltages of $V_{\text{thresh}} = -48\,\text{mV}$ and $V_{\text{thresh}} = -49\,\text{mV}$ the network neurons start spiking continuously. When in simulation the threshold voltage is set to be smaller then the resting voltage $V_{\text{thresh}} < V_{\text{rest}}$, we can observe the exact same behaviour as explained in section 3.2 and shown in figure 3.3d. In both, simulation and hardware emulation, the resting voltage was set to be $V_{\text{rest}} = -50\,\text{mV}$. As both, V_{thresh} and V_{rest}, have a relative deviation of around 1% and 9% (*Schmidt*, 2014), respectively, for some neurons both voltages will be close enough to initiate continuous spiking. The connectivity in the network will then result in other neurons receiving continuous input followed by again continuous spiking, and so on. This explains why we observe the effect already for $V_{\text{thresh}} = -48\,\text{mV}$.
For $V_{\text{thresh}} = -46\,\text{mV}$ (figure 3.4c) and $V_{\text{thresh}} = -47\,\text{mV}$ (figure 3.4d) we can observe, how the neurons of a population spike only a few times, transporting the signal from one population to the next. We therefore reproduced similar spiking patterns as the simulations in figures 3.3b and 3.3c showed. The main difference to the simulations is that we can not observe a significant increase of spikes per population for populations coming later in the chain. At the same time it was not possible to produce a spiking pattern where neurons of each population spike only once.
Both can be explained with the variation of hardware parameters. A setup that in simulation would result in exactly one spike per neuron as can be seen in figure 3.3a is very likely to result in a break up of the signal transportation in a hardware emulation as can be seen in figure 3.4a and 3.4b. Exactly 4 input spikes result in a rise in membrane potential of a receiving neuron that triggers exactly one spike in that neuron, in simulation. When connecting only 3 neurons, the receiving neuron might, depending on the other parameters, not spike anymore. Considering hardware parameter variations, we can assume that in a setup which should result in each neuron to spike only once, some hardware neurons might not spike even when receiving 4 input spikes. If such a neuron is one of the four neurons connected to the next population, the signal forwarding could already be stopped as the entire next population only receives three input spikes. This is just a qualitative example that should make clear how hardware parameter variations result in breaking of the spike chain.
If a neuron would receive two input spikes at the same time in simulation, on the hardware those two inputs would be sent just behind each other with a little delay. Therefore two spikes that would in simulation reach a neuron at the exact same time and increase the membrane potential of the receiving neuron just enough to trigger a spike, might not do so on hardware as the delay between the two spikes would leave room for membrane potential to decrease again towards its resting potential. Populations coming later in the chain network receive several spikes in short time from 4 sources (in simulation these inputs would arrive at the exact same time). Therefore

it is possible that some of the spiking in simulation is lost due to spike delays on the hardware. The delay effect is very small though and it would need some further investigation to determine if it actually has any affect.
Additionally the output buffers on the HICANN are capable to only register a limited amount of data. In a population coming later in the chain network every neuron will produce several spikes. This possibly could reach a limit in the output buffer resulting in not all spikes being registered.
In figure 3.4e we observe a surprising behaviour. In each of 4 hardware emulations with $V_{\text{thresh}} = -48\,\text{mV}$ the second input from the spike source array resulted in a stop of all spiking in the network. The same could be observed for 2 out of 4 emulations with $V_{\text{thresh}} = -49\,\text{mV}$. This could be either a bug in the spike recording mechanism or in the analog neuron circuit. To determine weather the spikes are just not recorded correctly or do not occur at all an analog voltage trace should be recorded. As at the time of writing this report the voltage recording function of the PyHMF module (PyHMF.Population.record_v()) seems to only record noise around 0.08 mV instead of the actual membrane potential[2], it was not possible to investigate this further.

3.4 Summary

In this internship a simple network model (section 3.1) was simulated and the spiking behaviour of neurons of different populations was investigated for different values of V_{thresh}. The different spiking behaviour was analyzed and explained and similar spiking patterns could be reproduced in hardware emulations on the HICANNv2 wafer.
Not entirely clear was the effect that continuous spiking in the network was interrupted by a single input spike from the spike source to the neurons of the first populations (figure 3.4e). To understand the source of this behaviour, an analog voltage trace needs to be recorded and more detailed experiments could be useful to investigate under which circumstances this behaviour is most likely to occur. More emulations for smaller steps of different threshold voltages in the range of $-49\,\text{mV} \leq V_{\text{thresh}} \leq -47\,\text{mV}$ could be done to find out if this effect really is related to the value of V_{thresh}, which cannot be concluded from the few experiments taken in this internship.

[2] `https://brainscales-r.kip.uni-heidelberg.de/issues/1567`

4 Outlook

After gaining necessary knowledge and skills to run simulations and hardware experiments it is planned to implement a more complicated network model on the HICANNv2 wafer. The chosen network is a modular network model of the cerebral neocortex first suggested by *Lundqvist et al.* (2006) and later adapted to the hardware limitations of the HICANNv2 by *Petrovici et al.* (2014).
To investigate the behaviour of single cells of the network on hardware, typical spike inputs of network neurons have already been recorded from the network simulation[1]. For recording single cells input the pyrec program[2] written by Paul Müller was used. It is planned to implement those neurons with the recorded inputs on the HICANNv2 and try to receive similar output as in the simulation. After that the network is planned to be build up step by step starting with smaller subunits and continuing towards implementing a downscaled version of the entire network used by *Petrovici et al.* (2014).

[1] `https://brainscales-r.kip.uni-heidelberg.de/projects/hbp-sp9-benchmark-model-kthl23`
[2] `https://brainscales-r.kip.uni-heidelberg.de/projects/hicann-dls-modeling`

Appendix

A: Simulation Parameters

Parameter	**Value**
C_{m}	0.2 nF
τ_{m}	20.0 ms
τ_{refrac}	0.0 ms
$\tau_{\mathrm{syn,E}}$	5.0 ms
$\tau_{\mathrm{syn,I}}$	5.0 ms
$E_{\mathrm{rev,I}}$	-60 mV
$E_{\mathrm{rev,E}}$	-40 mV
V_{reset}	-70 mV
V_{rest}	-50 mV
V_{thresh}	varied
I_{offset}	0.0 nA
weight	0.016 nA
delay	0.1 ms

Table .1: Parameters used in PyNN. Neurons were simulated with IF_cond_exp model and NEST backend.

B: Hardware Emulation Parameters

Parameter	**Value**
C_{m}	0.2 nF
τ_{m}	409.0 ms*
τ_{refrac}	20.0 ms*
$\tau_{\mathrm{syn,E}}$	5.0 ms
$\tau_{\mathrm{syn,I}}$	5.0 ms
$E_{\mathrm{rev,I}}$	-60 mV
$E_{\mathrm{rev,E}}$	-40 mV
V_{reset}	-70 mV
V_{rest}	-50 mV
V_{thresh}	varied
I_{offset}	0.0 nA
weight	0.004 nA
delay	0.1 ms

Table .2: Parameters used in PyNN for emulation on HICANNv2 wafer. Neuron model used: IF_cond_exp.
*Note that τ_{m} and τ_{refrac} were taken from the original code by Sebastian Schmitt where they were passed to the hardware directly as DAC values (hack in calibtic software). When the code was rerun, this hack was not used but instead the calibration software set the corresponding hardware parameters to to the fixed values of $I_{\mathrm{gl}} = 409\,\mathrm{DAC}$ and $I_{\mathrm{pl}} = 100\,\mathrm{DAC}$ for any values of τ_{m} and τ_{refrac}, respectively.

Bibliography

Gerstner, W., W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*, Cambridge University Press, 2014.

HBP SP9 partners, *Neuromorphic Platform Specification*, Human Brain Project, 2014.

Jeltsch, S., A scalable workflow for a configurable neuromorphic platform, Ph.D. thesis, Universität Heidelberg, 2014.

Lundqvist, M., M. Rehn, M. Djurfeldt, and A. Lansner, Attractor dynamics in a modular network of neocortex, *Network: Computation in Neural Systems*, *17:3*, 253–276, 2006.

Petrovici, M. A., et al., Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms, *PLOS ONE*, doi:dx.doi.org/10.1371/journal.pone.0108590, 2014.

PyNN, *A Python package for simulator-independent specification of neuronal network models*, The NeuralEnsemble Initiative, [Online; accessed: 2014-04-29], 2014.

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