Fakultät für Physik und Astronomie
Universität Heidelberg

Bachelorarbeit in Physik
eingereicht von

Binh Anton Tran

geboren in Reutlingen (Deutschland)

2013

Demonstrationsexperimente auf neuromorpher Hardware

Diese Bachelorarbeit wurde von Binh Anton Tran am Kirchhoff-Institut für Physik in Heidelberg geschrieben, unter der Aufsicht von
Prof. Dr. Karlheinz Meier

Zusammenfassung

In dieser Arbeit wird eine Software vorgestellt, die es erlaubt grundlegende und anschauliche Experimente auf einem neuromorphen Chip durchzuführen. Die Experimente sind aus einer graphischen Benutzeroberfläche heraus zu steuern, in der 512 unterschiedliche Neuronen auf verschiede Art und Weisen stimuliert werden können. Die dadurch entstehenden Spikes und postsynaptischen Potentiale sind für das Verständnis der Informationsübertragung im menschlichen Nervensystem von fundamentaler Bedeutung und können aus der Software heraus dargestellt werden. Weiterhin zeigen die Neuronen, die in der neuromorphen Hardware nach dem AdEx Neuronenmodell implementiert wurden, je nach Wahl ihrer Parameter eine Reihe von Feuermustern, die in dieser Arbeit präsentiert werden. Neben einfachen Feuermustern wie Tonic Spiking konnte für ein bestimmtes Parameterset leicht chaotisches Verhalten erzeugt werden.

Abstract

This thesis presents a software serving as a platform for basic and illustrative experiments on a neuromorphic chip. A graphical user interface is implemented in order to run experiments which include different ways of neuron stimulation. The resulting spikes and postsynaptic potentials are fundamental to information processing in the human brain and can be directly displayed in the presented software. Moreover, the neurons in the hardware which are implemented after the AdEx model, show several firing patterns depending on the choice of parameters. These firing patterns are reproduced and described in this work and include simple patterns such as tonic spiking as well as chaotic ones.

Inhaltsverzeichnis

1 Einleitung

Als Ramón y Cajal zusammen mit Camillo Golgi 1906 mit dem Nobelpreis für seine "Arbeit über die Struktur des Nervensystems"gewürdigt wurde, war dies ein Meilenstein in der Neurowissenschaft. Er erkannte, dass das menschliche Nervensystem aus vielen Millarden individuellen Zellen, den "Neuronen", besteht, die über kleine Kontaktstellen, den "Synapsen", miteinander verbunden sind. Obwohl die Struktur des Nervensystems seither immer detaillierter erforscht wurde, bleibt ein wesentlicher Aspekt des Gehirns noch weitgehend unverstanden: die Informationsverarbeitung. Sie besticht u.a. durch einen extrem geringen Energieverbrauch, sowie eine hohe Fehlertoleranz. Auch bei Entscheidungsfindungen oder Lernprozessen ist das menschliche Gehirn jedem modernen Computer überlegen.
Eine wesentliche Rolle in der Informationsübertragung des Nervensystems spielen elektrische Signale. Wird ein Neuron durch einen Strom angeregt, reagiert dieses durch Aussenden eines *Spikes* (Aktionspotential), einer kurzfristigen Änderung seiner Membranspannung (in der Neurowissenschaft auch häufig als Membranpotential bezeichnet).
Im Rahmen des BrainScaleS Projekts wird versucht, die Prozesse der Informationsverarbeitung mithilfe von neuromorpher Hardware zu emulieren. Da es zurzeit unmöglich ist, das Gehirn in seiner vollständigen Komplexität nachzuahmen, wurden Neuronenmodelle wie das "Adaptive Exponential Integrate-and-Fire Modell"(Brette & Gerstner 2005, [1]) entwickelt. Dieses Modell besteht aus zwei nichtlinearen gekoppelten Differentialgleichungen, die in der Hardware durch analoge Schaltkreise implementiert werden können. Der große Vorteil dieser Vorgehensweise im Gegensatz zu Simulationen durch Computer besteht in der hohen Geschwindigkeit, welche auch die Untersuchung von Lern- und Entwicklungsprozessen ermöglicht.
Die vielversprechenden Ziele des BrainScaleS Projekts erweckten in den letzten Jahren auch immer mehr das öffentliche Interesse, insbesondere stieg die Nachfrage nach einer geeigneten Möglichkeit, die Funktionen der neuromorphen Hardware zu demonstrieren. Daher ist das Ziel dieser Arbeit eine leicht zu bedienende Software zu entwickeln, welche die Funktionsfähigkeit der Hardware in einer anschaulichen Weise darstellen soll. Desweiteren stellt die Software eine leicht erweiterbare Plattform für Hardwaretests dar, die unkompliziert über eine graphische Oberfläche durchgeführt werden können.

2 Neuronenmodelle

2.1 Das Leaky Integrate-and-Fire Modell

Eines der einfachsten und am weit verbreitesten Neuronenmodelle ist das Leaky Integrate-and-Fire Modell (LIF) [2]. Der zugrundeliegende Schaltkreis besteht aus einem Kondensator C (Membran des Neurons) und einem dazu parallel geschalteten Widerstand R (Leck, engl. "leakage"), welcher von einem äußeren Strom I angetrieben wird. Der Schaltkreis beschreibt also im Wesentlichen das Laden eines Kondensators und kann durch folgende Gleichung dargestellt werden:

$$C\frac{dV}{dt} = -g_L(V - E_L) + I \tag{1}$$

Dabei bezeichnen V die Membranspannung, $g_L = 1/R$ die Leckkonduktanz und E_L die Leckspannung. Erreicht die Membranspannung $V(t)$ eine gegebene Schwellenspannung V_T, so wird die Membranspannung auf die Spannung V_{reset} zurückgesetzt :

$$V \to V_{reset} \tag{2}$$

2.2 Das Adaptive Exponential Integrate-and-Fire Modell

Das Adaptive Exponential Integrate-and-Fire Modell (AdEx) ist ein erweitertes Neuronenmodell, das 2005 von Romain Brette und Wulfram Gerstner eingeführt wurde. Dieses Modell ist in der vorliegenden neuromorphen Hardware realisiert und wird im Gegensatz zum LIF-Modell u.a. durch eine zusätzliche Adaptation w beschrieben:

$$C\frac{dV}{dt} = f(V) + w + I \tag{3}$$

Die Funkiton $f(V)$ charakterisiert den Spike-Mechanismus und setzt sich aus einem linearen und einem exponentiellen Term zusammen:

$$f(V) = -g_L(V - E_L) + g_L\Delta_T \exp(\frac{V - V_T}{\Delta_T}) \tag{4}$$

Hinzu kommt eine zweite Differentialgleichung, welche die Adaptation beschreibt:

$$\tau_w\frac{dw}{dt} = a(V - E_L) - w \tag{5}$$

Hierbei bezeichnen Δ_T den Steigungsfaktor, τ_w die Adaptationszeitkonstante und a die unterschwellige Adaptationskonduktanz. Analog zum LIF-Modell gibt es hier zwei Reset-Bedingungen:

$$V \to V_{reset} \tag{6}$$

$$w \to w + b \tag{7}$$

3 Experimenteller Aufbau

3.1 Hardwareplattform

Die meiste Zeit dieser Arbeit wurde auf einer Hardwareplattform gearbeitet, die als Prototyp eines Retikels des Wafer-Scale Systems (siehe [15]) dient. Diese besteht u.a. aus einem FPGA[1]-Board, das über Ethernet mit einem PC verbunden ist, einem DNC[2]-Board und einem SEB[3]. Am SEB sind bis zu 8 HICANN[4]-Chips angeschlossen, sowie zwei ADCs[5], die das Auslesen der Membranspannung ermöglichen. Die HICANN-Chips dienen der Emulation von Neuronen und werden in 3.2 erläutert. Eine detailliertere Beschreibung des HICANN-Chips ist z.B. in [6] zu finden.

3.2 HICANN-Chip

Der HICANN-Chip lässt sich in vier Komponenten unterteilen:

- **Neuronenblock:** Zwei Reihen mit jeweils 256 Dendrit Membran (DenMem) Schaltkreisen bilden die Neuronen nach. Abb. 2 stellt einen Denmem Schaltkreis dar. Er ist in viele untergeordnete Schaltkreise unterteilt (grün), die nach dem AdEx-Modell jeweils die unterschiedlichen mathematischen Terme repräsentieren.

[1]Field Programmalble Gate Array

[2]Digital Network Chip

[3]System Emulator Board, entwickelt in der Electronic Vision(s) Arbeitsgruppe an der Universität Heidelberg

[4]High Input Count Analog Neural Network

[5]Analog to Digital Converter

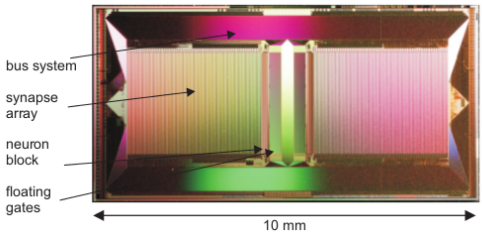

Abbildung 1: Aufbau eines HICANN-Chips. Abbildung entnommen aus [14]

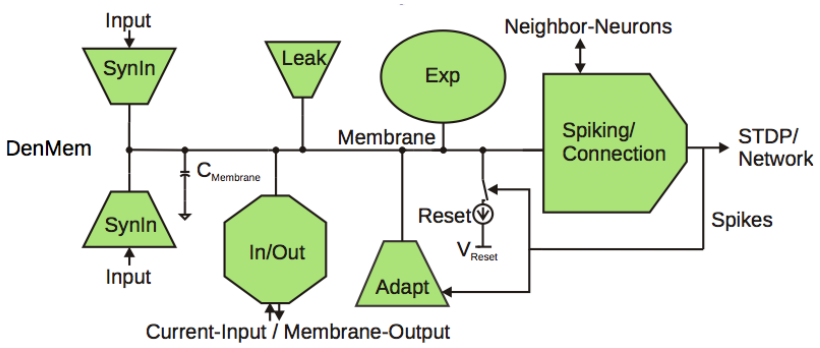

Abbildung 2: Schaubild eines DenMem Schaltkreis. Abbildung aus [4]

- **Synapsenarray:** Ein Synapsenarray besteht aus 224 x 256 Synapsenschaltkreisen. Unterhalb und oberhalb eines Synapsenarrays befinden sich je 56 Synapsentreiber, welche die Schnittstelle zwischen den Eventdaten und den Synapsenarrays bilden.

- **Floating Gates:** Die Floating Gates dienen der Speicherung von Neuronparametern, welche gemäß dem AdEx-Modell das Verhalten des Neurons bestimmen. Diese Parameter werden durch 12 einstellbare Spannungen und 9 Ströme realisiert und können in vier Floating Gate Arrays gespeichert werden. Jedes Array besteht dabei aus 24 Zeilen mit jeweils 129 Speicherzellen. Neben Neuronparametern werden in der 0.Zeile globale Parameter gespeichert, welche neuronübergreifenden Zwecken dienen.

- **Bussystem:** Das Bussystem dient der Kommunikation zwischen Neuronen. Man unterscheidet dabei zwischen dem Layer 1 (L1) Protokoll, welches primär der intra-Wafer-Kommunikation dient und dem Layer

2 (L2) Protokoll, das z.B. die inter-Wafer-Kommunikation ermöglicht.

4 Demonstrationssoftware

4.1 Verwendete Sofware

Die vorliegende Software wurde in der Programmiersprache Python 2.7 [7] implementiert. Für die graphische Benutzeroberfläche (kurz: GUI[6]) wurde die Klassenbibliothek Qt bzw. die dazugehörige Anbinding pyQt verwendet [8].Um eine einfache und schnelle Erweiterbarkeit der GUI zu ermöglichen, wurde der WYSIWYG[7]-Editor Qt Designer benutzt [8]. Plots sind mithilfe der Pythonbibliothek matplotlib dargestellt [9].
Zur Programmierung wurde auf gruppeninterne Funktionsbibliotheken (Halbe, pyHMF, calibtic; siehe [10]) zurückgegriffen, die u.a. den Zugriff auf die Hardware ermöglichen.

4.2 Überblick

Die Neuron Demo setzt sich aus verschiedenen Komponenten (Widgets) zusammen, die jeweils abgeschlossene Funktionseinheiten darstellen (vgl. Abb. 3).

- Setup Einstellungen
- Schreiben von Floating Gate Parametern
- Stimulus eines Neurons durch externen Strom
- Stimulus eines Neurons durch Synapse

Eine genaue Beschreibung der Komponenten erfolgt in Abschnitt 4.3

4.3 Komponenten

4.3.1 Setup

Dieses Widget dient dazu, die Neuron Demo für verschiedene Hardwaresetups konfigurierbar zu machen. Da die Kommunikation zwischen PC und Setup

[6]Graphical User Interface
[7]What You See Is What You Get

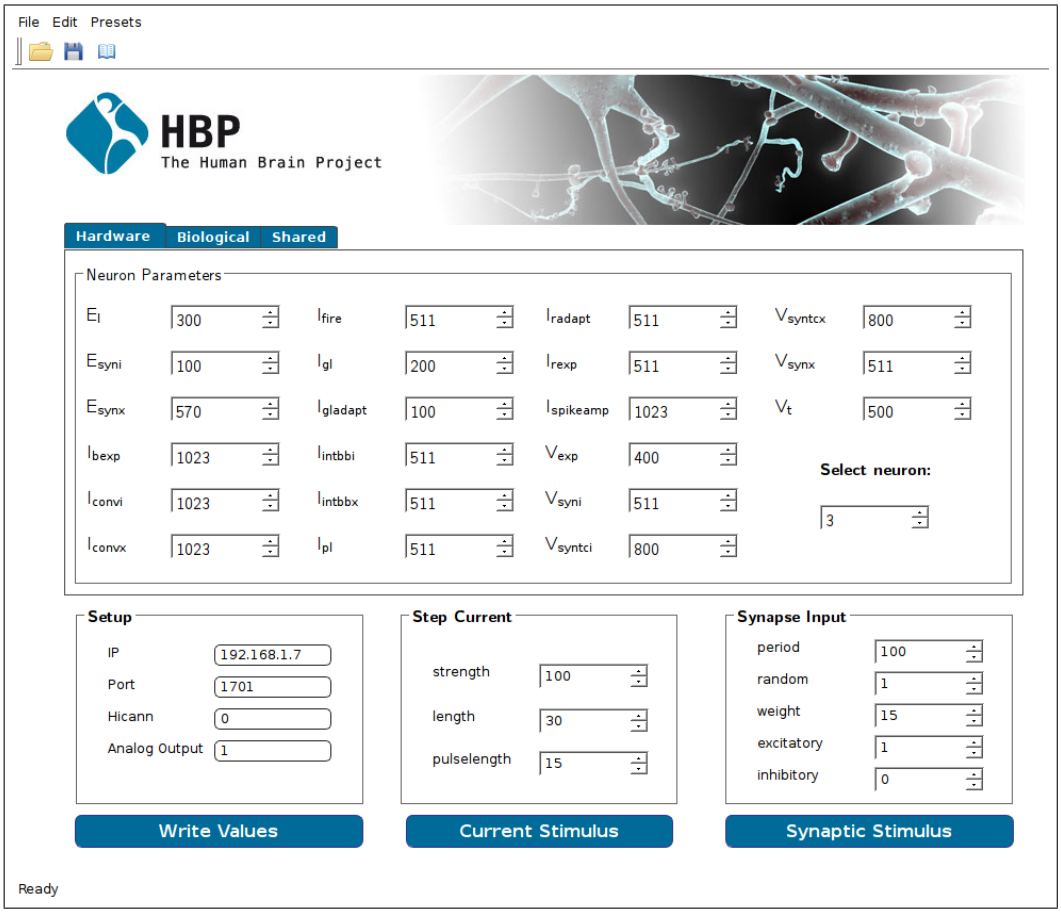

Abbildung 3: Hauptfenster der Neuron Demo

über Ethernet erfolgt, ist eine IP-Adresse anzugeben, um das gewünschte FPGA-Board anzusprechen. Über die Portnummer erfolgt die Zuordnung eines Retikels und schließlich kann innerhalb dieses Retikels der HICANN-Chip ausgewählt werden. Zuletzt kann noch bestimmt werden, über welchen Ausgang des ADC die Membranspannung ausgelesen werden soll.

4.3.2 Schreiben von Floating Gate Parametern

Beim Starten der Neuron Demo werden zunächst die in pyHALbe implementierten default Parameter in eine interne Datenstruktur geladen. Mithilfe von SpinBoxes kann der Benutzer dann 21 Neuronparameter, sowie 22 globale

Parameter einstellen, welche sich jeweils in einem Wertebereich von 0-1023 (digitale Werte) erstrecken. Die Bedeutung der verschiedenen Parameter ist im Anhang A zu finden. Der Zugriff auf die Floating Gates erfolgt über einen digitalen Controller (entwickelt von S.Millner [12]), der für die Parameterprogrammierung, das analoge Auslesen von Floating Gate Zellen und für die Neuronenstimulierung zuständig ist.
Der Button *Write Values* erstellt zunächst eine Instanz dieses Floating Gate Contollers und liest sämtliche Neuronparameter, globale Parameter und Setupeinstellungen aus der Datenstruktur aus. Der eigentliche Programmiervorgang der Parameter im Floating Gate Array erfolgt dann zeilenweise und in eine Richtung. In jeder Zeile werden die einzelnen Parameter jeder Zelle mit den gewünschten Parametern verglichen, die im RAM des Controllers gespeichert werden. Stimmt der Wert in einer Zelle bereits mit dem gewünschten Wert überein, wird die Zelle ausgelassen, ansonsten an den Wert angepasst. Da innerhalb einer Zeile immer nur eine Erhöhung oder nur eine Erniedrigung der Werte möglich ist, erfolgt der Programmiervorgang in zwei Richtungen, zunächst mit einem writeDown Befehl und dann mit einem writeUp Befehl. Es wird solange verglichen und geschrieben bis alle Parameter den gewünschten Wert erreicht haben oder bis eine festgelegte Anzahl an Schreibzyklen erreicht ist [6].
Eingestellte globale und Neuronparameter können schließlich in einer .json[8]-Datei gesichert werden, um Messergebnisse zu beliebiger Zeit reproduzieren zu können. Um auch weniger hardwareversierten Benutzern eine Arbeitsplattform zu bieten, können anstelle von digitalen Werten für die Neuronparameter ebenso biologische Parameter eingestellt werden. Da die biologischen Parameter auf der Hardware beschränkt sind und je nach Parameter andere Wertebereiche aufweisen (vgl. [3] S.73), wurden zur Bedienung hier Slider gewählt, die eine intuitivere Steuerung zulassen. Auch hier werden beim Starten des Programms zunächst die in pyHMF vorhandenen default-Parameter eingestellt. Mit dem Verstellen eines Sliders erfolgt zeitgleich eine Umrechnung (nach [3], S.66 ff.) der biologischen Parameter in Hardwareparameter. Ein Zusammenhang zwischen diesen beiden Parameterdarstellungen kann Tab. 1 entnommen werden.

[8] JavaScript Object Notation

a	$I_{gladapt}$	Unterschwellige Adaptationskonduktanz
b	I_{fire}	Adaptation ausgelöst durch Spikes
Δ_T	I_{rexp}	Steigungsfaktor des Exponentialterms
E_{RevE}	E_{synx}	Synaptisches Umkehrpotential (exzitatorisch)
E_{RevI}	E_{syni}	Synaptisches Umkehrpotential (inhibitorisch)
τ_{refrac}	I_{pl}	Refraktärzeit
τ_{synE}	V_{syntcx}	Zeitkonstante der synaptischen Konduktanz (exzitatorisch)
τ_{synI}	V_{syntci}	Zeitkonstante der synaptischen Konduktanz (inhibitorisch)
$\tau_m = C_m/g_L$	I_{gl}	Membranzeitkonstante, prop. zur Leckkonduktanz
τ_w	I_{radapt}	Adaptationszeitkonstante
V_{reset}	V_{reset}	Rückstellwert der Membranspannung nach Spike
V_{rest}	E_L	Ruhemembranpotential
V_{spike}	V_t	benötigte Membranspannung um Spike zu detektieren
V_{thresh}	V_{exp}, I_{exp}	Schwellspannung des Exponentialterms

Tabelle 1: Zusammenhang zwischen biologischen und Hardware Parametern

4.3.3 Stimulus durch externen Strom

Durch Betätigen des Buttons *Current Stimulus* wird das ausgewählte Neuron mit einem periodischen Rechteckstrom stimuliert. Dabei können Stärke (0-1023), Länge(0-129) und Pulslänge des Stroms variiert werden werden. Die Länge gibt dabei die Anzahl der einzelnen Pulse an. Als Stromquelle für die Neuronenstimulation dient ein speziell für großskalige neuromorphe Systeme von S. Millner entwickelter Schaltkreis (vgl. Abb.4).

Die zugrundeliegende Spannung V_{ref} wird dabei durch einen DAC[9] erzeugt, der in den Floating Gate Arrays implementiert ist. Jedes Floating Gate Array wird von einem digitalen Controller gesteuert, der dafür sorgt, dass die Spannung zeitlich variiert werden kann. Der Strom, welcher mit dem Spannungsabfall am Widerstand R korrespondiert, wird über eine Stromspiegelschaltung schließlich auf das Neuron gegeben. Zuletzt wird der zeitliche Verlauf der Membranspannung über ein ADC Board ausgelesen und in einem matplotlib Fenster dargestellt.

[9]Digital-To-Analog Converter

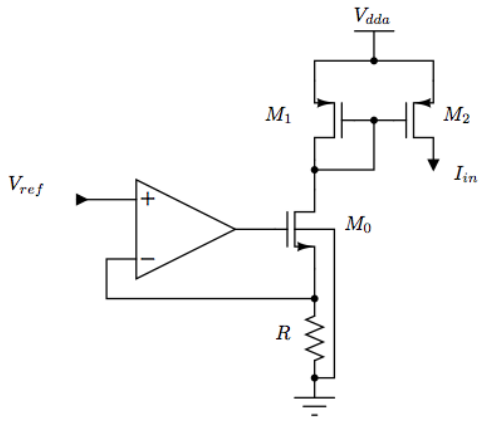

Abbildung 4: Stromquelle für Neuronenstimulation. Abbildung entnommen aus [12]

4.3.4 Stimulus durch Synapsen

Alternativ zur Anregung durch einen externen Strom lassen sich die Neuronen auch durch Synapsen stimulieren (Button: *Synaptic Stimulus*). Das Event kommt von einem Background Event Generator, einem Schaltkreis der je nach Einstellung periodische oder zufällige Spiketrains erzeugt. So bedeutet random = 0, dass eine feste Periode zwischen den Events festgelegt wird, wohingegen die Intervalle der Events bei random = 1 poissonverteilt sind. In letzerem Fall gibt der Parameter *period* das mittlere ISI[10] an und bei fester Periode die Anzahl der Taktzyklen zwischen zwei Events [6]. Weiterhin kann ausgewählt werden, ob die Synapse exzitatorisch (d.h. stimulierend) oder inhibitorisch (hemmend) auf das postsynaptische Neuron wirken soll und welches Gewicht (0-15) die Synapse hat. Diese Parameter werden über einen digitalen Synapsen Controller gesteuert. In Abb. 5 - 7 sind einige aufgenommene Beispiele von postsynaptischen Potentialen (PSP) zu sehen, die eine typische α-Form besitzen. In Abb.5 sind exzitatorische postsynaptische Potentiale (EPSP) dargestellt, die durch poissonverteilte Events des Background Generators erzeugt werden. Das ISI zwischen den PSPs ist wie erwartet unterschiedlich, und auch die Amplitude der PSPs ist nicht immer identisch. Die Abweichung der Amplitude ist durch eine Überlagerung von

[10]inter-spike interval

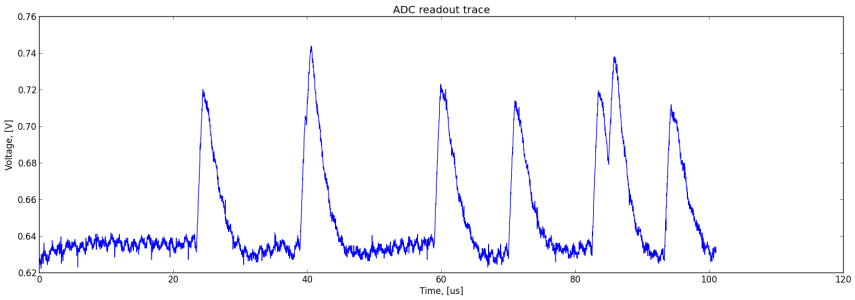

Abbildung 5: Exzitatorische postsynaptische Potentiale bei poissonverteilten Events

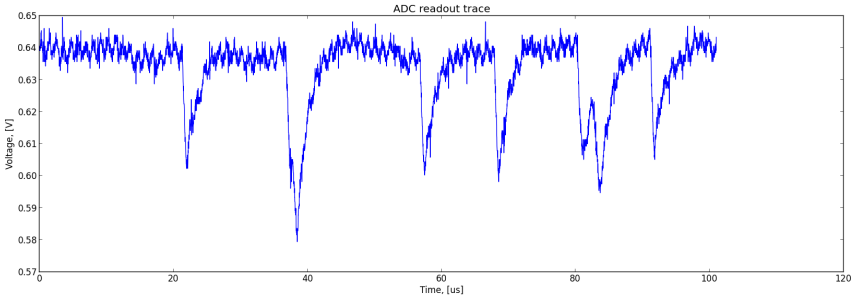

Abbildung 6: Inhibitorische postsynaptische Potentiale bei poissonverteilten Events

mehreren PSPs zu erklären. Abb. 6 zeigt das gleiche Verhalten für inhibitorische synaptische Stimulierung. Bei periodischen Events (Abb.7) zeigen die PSPs dagegen ein konstantes ISI, sowie identische Amplituden.

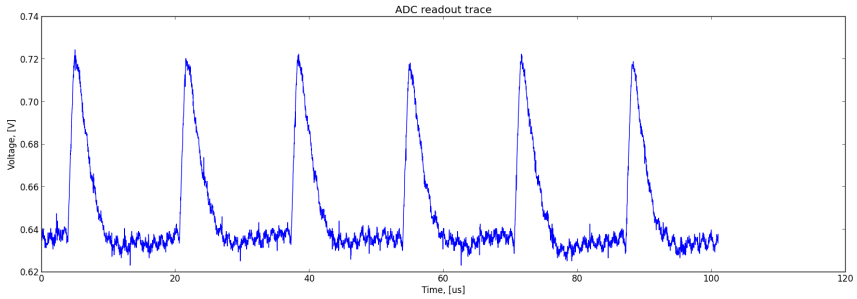

Abbildung 7: Exzitatorische postsynaptische Potentiale bei periodischen Events

5 Feuermuster des AdEx Modells

Abhängig von der Wahl der Parameter zeigt das AdEx Modell eine Reihe von Feuermustern, die in [11] detailliert beschrieben sind und im Folgenden reproduziert werden sollen. Dabei werden die Neuronen mit einem Rechtecktrom stimuliert.
Um die Anzahl der Parameter zu reduzieren, werden die Differentialgleichungen (3) und (5) des AdEx-Modell auf eine Form gebracht, in der nur noch vier freie Parameter (Bifurkationsparameter) und ein Stromterm existieren (nach [13]):

$$\frac{d\bar{V}}{d\bar{t}} = -\bar{V} + e^{\bar{V}} - \bar{w} + \bar{I} \tag{8}$$

$$\bar{\tau_w}\frac{d\bar{w}}{d\bar{t}} = \bar{a}\bar{V} - \bar{w} \tag{9}$$

Resetbedingungen:

$$\bar{V} \to \bar{V}_{reset}$$
$$\bar{w} \to \bar{w} + \bar{b}$$

Dabei sind:

$$\bar{\tau}_w := \frac{\tau_w}{\tau_m} = \frac{g_L \tau_w}{C}$$
$$\bar{a} := \frac{a}{g_L}$$
$$\bar{I} := \frac{I}{g_L \Delta_T} + (1 + \frac{a}{g_L}) \frac{E_L V_T}{\Delta_T}$$
$$\bar{t} := \frac{t}{\tau_m}$$
$$\bar{b} := \frac{b}{g_L \Delta_T}$$
$$\bar{V}_{reset} := \frac{V_{reset} - V_T}{\Delta_T}$$
$$\bar{V}(\bar{t}) := \frac{V(t) - V_T}{\Delta_T}$$
$$\bar{w}(\bar{r}) := \frac{w(t) + a(E_L - V_T)}{g_L \Delta_T}$$

Die AdEx Gleichungen wurden auf eine dimensionlose Form gebracht, indem z.B. die Zeit t auf Einheiten der Membranzeitkonstante τ_m skaliert wurde oder die unterschwellige Adaptationskonduktanz a auf Einheiten der Leckkonduktanz g_L. Hält man nun die Skalierungsparameter konstant, sind die vier Bifurkationsparameter direkt proportional zu den AdEx Parametern a, b, τ_w und V_{reset} (bzw. in Hardware $I_{gladapt}$, I_{fire}, I_{radapt} und V_{reset}), sodass eine Variation dieser Parameter auch eine Änderung am gesamten System bewirkt. Die Parametersätze, die zur Erzeugung der folgenden Feuermuster auf der Hardware verwendet wurden, sind im Anhang C zu finden.

5.1 Tonic Spiking

Tonic Spiking ist das einfachste Feuermuster und kann generell auch mit dem LIF Modell produziert werden. Bei Stimulation mit einen Strom zeichnet es sich durch ein Feuern mit einer konstanten Spikefrequenz aus. Im AdEx Modell erreicht man dies durch das Abstellen des Adaptationsterms ($I_{gladapt} = I_{fire} = I_{radapt} = 0$) bei einem ausreichend hohen Strom. Dieses Verhalten wurde mithilfe der Neuron Demo auf dem HICANN-Chip reproduziert und ist in Abb.8 dargestellt.

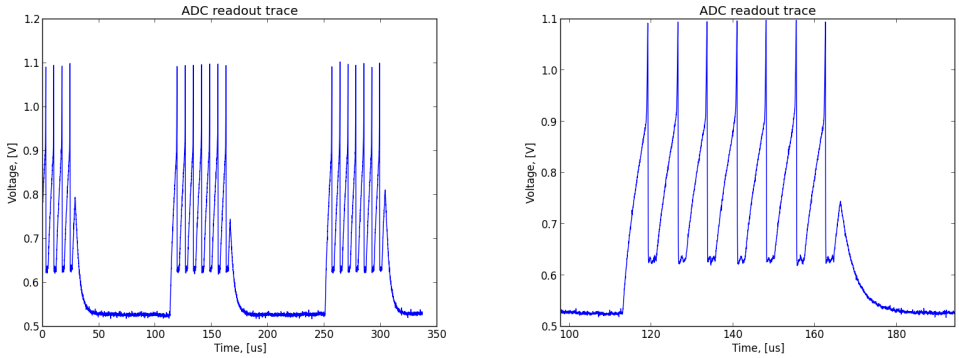

Abbildung 8: Links: Tonic Spiking, Rechts: vergrößerter Ausschnitt

5.2 Spike Frequency Adaptation

Dieses Feuermuster ist im Gegensatz zu Tonic Spiking nur im AdEx-Modell realisierbar und ist dadurch charakterisiert, dass das ISI während dem Stimulus immer stärker anwächst.

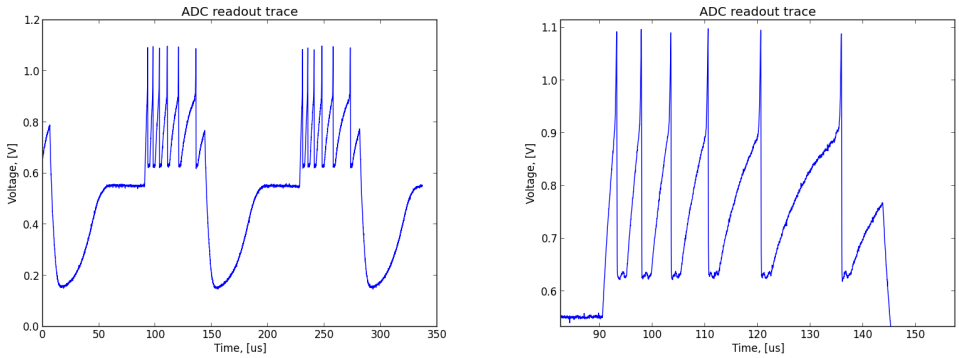

Abbildung 9: Links: Spike Frequency Adaptation, Rechts: vergrößerter Ausschnitt

Hierfür wurden die Parameter $I_{gladapt}$, I_{fire} und I_{radapt} wieder aktiviert. Der Adaptationsprozess ist gut zu sehen, wenn I_{fire} kleiner als die beiden anderen Parameter gewählt wird, da die zeitliche Änderung der Adaptation $\frac{dw}{dt}$ dann nur langsam abnimmt. Gleichzeitig muss I_{fire} noch so groß sein, dass sich die Adaptation $w \to w + b$ auf ein sichtbares Maß erhöht.

5.3 Phasic Spiking

Geht man von dem Feuermuster Spike Frequency Adaptation aus und verringert den Strom, beobachtet man, dass das Neuron nur einmal feuert und schließlich eine konstante Membranspannung annimmt (s. Abb.10). Dieses Verhalten ist ebenfalls nur im AdEx-Modell zu sehen. Im LIF Modell würde man hingegen erwarten, dass das Neuron entweder mit einer konstanten Frequenz oder überhaupt nicht feuert.

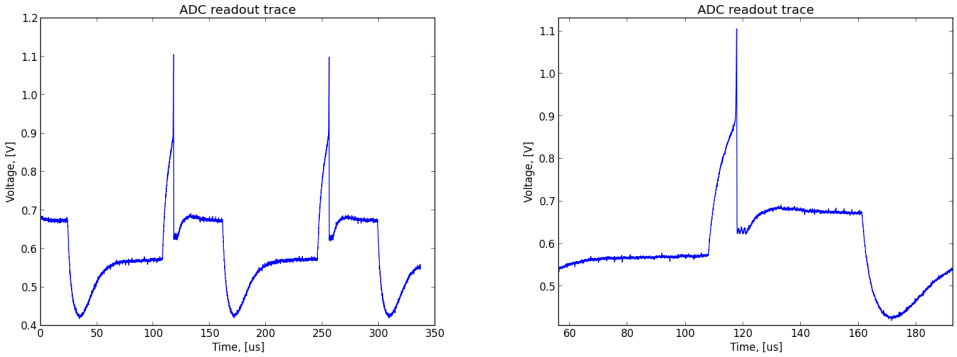

Abbildung 10: Links: Phasic Spiking, Rechts: vergrößerter Ausschnitt

5.4 Initial Bursting

Initial Bursting (manchmal auch als *mixed mode* bezeichnet) ist in erster Linie dadurch zu erkennen, dass das ISI zu Beginn während den ersten Spikes deutlich kleiner ist als bei den darauf folgenden Spikes, die eine konstante Frequenz annehmen. R. Naud [11] gibt eine etwas andere Definition, die es einfacher machen soll, Initial Bursting von einem ausgeprägten Adaptationsverhalten zu unterscheiden. Demnach ist Initial Bursting eine "geordnete Folge von scharfen hin zu breiten Resets (z.B. scharf-scharf-breit-breit-breit)". Breite Resets zeichnen sich im Gegensatz zu scharfen Resets durch eine höhere Krümmung aus, wobei eine bessere Unterscheidung durch Phasenraumanalysen erfolgen kann (Durchqueren der V-Nullkline mit Vorzeichenwechsel von $\frac{dV}{dT} < 0$ nach $\frac{dV}{dT} > 0$). Nichtsdestotrotz ist der Unterschied in Abb. 11 gut zu erkennen: Nach einer Folge von fünf Spikes mit scharfen Resets und kleinem ISI, folgen weitere Spikes, deren Resets eine höhere Krümmung aufweisen und deren ISI auf einen größeren konstanten Wert gestiegen ist.

Um dieses Feuermuster auf der Hardware zu realisieren, wurden die beiden Parameter $I_{gladapt}$ und I_{radapt} auf zwei konstante Werte fixiert. In numerischen Untersuchungen des AdEx Modells [11] zeigte sich, dass für hohe Werte dieser beiden Parameter, das Volumen des Burstingmusters im Phasenraum von I_{fire} und V_{reset} besonders groß ist. Somit wurden I_{fire} und V_{reset} schrittweise für ein konstantes Wertepaar von $I_{gladapt}$ und I_{radapt} variiert, bis das in Abb.11 gezeigte Verhalten zu sehen war.

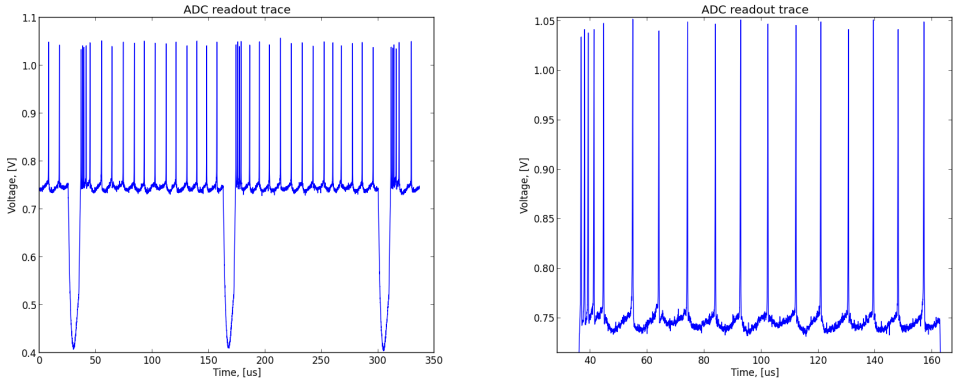

Abbildung 11: Links: Initial Bursting, Rechts: vergrößerter Ausschnitt

5.5 Phasic Bursting

Ähnlich wie bei Phasic Spiking kann dieses Feuermuster ausgehend von Initial Bursting durch ein Vermindern der Stromstärke erzeugt werden. Hierbei wird durch den Strom nur ein einziger Burst zu Beginn des Stimulus produziert (Abb.12).

5.6 Irregular Spiking/Chaos

Die in Abb. 13 dargestellte Aufnahme zeigt ein deterministisches Chaosverhalten im AdEx Modell. Es wird charakterisiert durch einen "Wechsel zwischen breiten und scharfen SAP[11], sodass die Anzahl der scharfen Resets zwischen den breiten Resets nach dem dritten breiten Reset nicht konstant ist"[11]. In Abb.13b) kann diese Definition bestätigt werden: Zwischen den

[11]Spike after potential

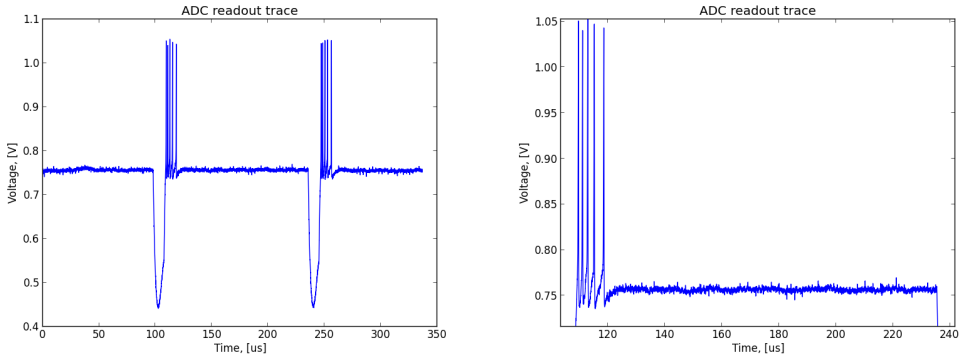

Abbildung 12: Links: Phasic Bursting, Rechts: vergrößerter Ausschnitt

breiten Resets ist bei etwa 113μs ein Burst mit 3 Spikes und scharfen Resets zu sehen, während bei etwa 130μs oder 180μs nach breiten Resets Gruppen von nur zwei Spikes mit scharfen Resets folgen. Auch der Vergleich von zwei verschiedenen Spiketrains in Abb.13a) zeigt keinerlei Periodizität. Dieses Spikeverhalten wurde ebenfalls durch ein schrittweises Variieren von I_{fire} und V_{reset} auf der Hardware realisiert.

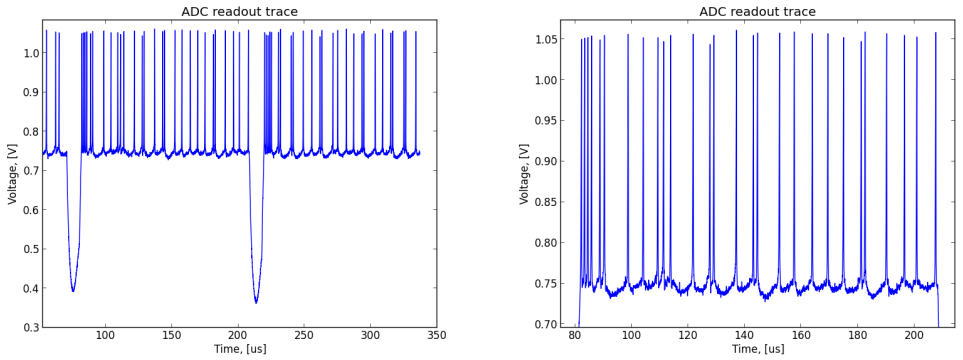

Abbildung 13: a) Links: Irregular Spiking, b) Rechts: vergrößerter Ausschnitt

5.7 Sonstige Feuermuster

Im AdEx-Modell sind noch weitere Feuermuster wie z.B. Delayed Accelerating denkbar. Diese erforden jedoch negative Adaptationsparameter, die im HICANN-Chip nicht zu realisieren sind ([3] S.73).

6 Zusammenfassung & Diskussion

Im ersten Teil dieser Arbeit wurde mit der Neuron Demo eine Software entwickelt, die in Zukunft dabei helfen soll, Experimente auf dem HICANN-Chip auf eine schnelle und einfache Weise durchzuführen, welche sowohl zu Demonstrations- als auch zu Forschungszwecken dienen können. Die Software ist grundsätzlich beliebig erweiterbar, sodass im nächsten Schritt z.B. Demos an einfachen neuronalen Netzwerken implementiert werden könnten.

Im zweiten Teil wurden einige aus dem AdEx Modell bekannten Feuermuster auf neuromorpher Hardware reproduziert und analysiert. Die vorliegenden Messungen und Aufnahmen bestätigen, dass das Neuronenmodell auf der Hardware auf eine hervorragende Weise repräsentiert wird. Erstmals konnte auf dem HICANN-Chip auch ein chaotisches Feuermuster erzeugt werden, was einmal mehr zeigt, dass auch komplizierte Sachverhalte des Neuronenmodells, welche sehr eingeschränkten Parameterbereichen unterliegen, auf dem Chip angemessen abgebildet werden.
Dennoch weist die neuromorphe Hardware auch einige systematische Fehlerquellen und Schwächen auf. Dazu gehört eine ungenaue Floating Gate Programmierung, die u.a. dazu führt, dass identische Parametersätze bei einer Stimulierung erhebliche Unterschiede im Spiking Verhalten vorzeigen können. Dies ist v.a. bei den Feuermustern Irregular Spiking und Phasic/Initial Bursting relevant, welche ohnehin sehr sensibel auf Parameteränderungen reagieren. Eine weitere Schwierigkeit entsteht durch den eingeschränkten Bereich einzelner Parameter auf der Hardware. So kann z.B. kein negativer Wert für die unterschwellige Adapation a erzeugt werden, und auch der Wert für den Parameter b ist nach oben hin limitiert (vgl.[3]). Da eine Änderung der Bifurkationsparamater auch eine Änderung der Volumina im Phasenraum für das Auftreten der verschiedenen Feuermuster bedeutet, ist dies ein weiterer erschwerender Faktor beim Erzeugen unterschiedlicher Verhaltensmuster.
Neben den eher qualitativen Analysen der Feuermuster in dieser Arbeit sind weitere quantitative Untersuchungen denkbar. Das Irregular Spiking Feuermuster könnte beispielsweise durch Erstellen eines Bifurkationsdiagramms oder eines Rekursionsdiagramms der ISI (ISI_{n+1} gegen ISI_n), wie in [11] vorgestellt, analysiert werden, welches zeigen würde, dass das Chaosverhalten deterministisch ist und nicht etwa durch elektronische Fehlerquellen erzeugt wurde.

Anhang

A Bedeutung der Neuronparameter

E_L	Leakage reversal potential
E_{syni}	Synapse inhibitory reversal potential
E_{synx}	Synapse excitatory reversal potential
I_{bexp}	Bias for operation amplifier of exponential circuit (tech.)
I_{convi}	Controls maximum inhibitory synaptic current
I_{convx}	Controls maximum excitatory synaptic current
I_{fire}	Current flowing onto adaptation capacitance at a spike
I_{gl}	Current that is proportional to leakage conductance for small signal and maximum leakage current
$I_{gladapt}$	Subthreshold adaptation conductance
I_{intbbi}	Bias for integrating operating amplifier in inhibitory synapse input (tech.)
I_{intbbx}	Bias for integrating operating amplifier in excitatory synapse input (tech.)
I_{pl}	Current for adjusting refractory period
I_{radapt}	Current for adjusting conductance in adaptation time constant
I_{rexp}	Exponential slope factor
$I_{spikeamp}$	Bias current of spike threshold comparator (tech.)
V_{exp}	Exponential reference potential
V_{syni}	Voltage controlling synapse excitatory reversal potential
V_{syntci}	Voltage controlling timeconstant of inhibitory synaptic pulse
V_{syntcx}	Voltage controlling timeconstant of exitatory synaptic pulse
V_{synx}	Zero voltage of collecting line from synapse array (tech.)
V_t	Membrane voltage needed to detect a spike

Tabelle 2: Bedeutung der Neuronparameter auf Hardware. Tabelle nach [10] und [6]. Da die englischen Fachbegriffe gebräuchlicher sind, wurde hier auf eine Übersetzung verzichtet.

B Bedeutung der globalen Parameter

I_{breset}	Current used to pull down membrane to reset potential
I_{bstim}	Bias for neuron stimulation circuit
int_op_bias	Internal OP bias
V_{bexp}	Bias for buffer of V_{exp}
V_{bout}	Global bias of neuron readout
V_{br}	Bias for STDP readout circuit
V_{bstdf}	Bias for short-term plasticity (STP)
V_{ccas}	Bias of L1 input amplifier
V_{clrc}	STDP CLR voltage (causal)
V_{clra}	STDP CLR voltage (acausal)
V_{dep}	Voltage used for short-term plasticity in depression mode
V_{dllres}	DLL reset voltage
V_{dtc}	Bias for DTC in short-term plasticity circuit
V_{fac}	Voltage used for short-term-plasticity in facilitation mode
V_{gmax0}	Sets synaptic conductance
V_{gmax1}	Sets synaptic conductance
V_{gmax2}	Sets synaptic conductance
V_{gmax3}	Sets synaptic conductance
V_m	Start load voltage of causal STDP capacitor (ground for acausal)
V_{reset}	Reset value for V_m after spike
V_{stdf}	STDF reset voltage
V_{thigh}	STDP readout compare voltage

Tabelle 3: Bedeutung der globalen Parameter auf Hardware. Tabelle nach [10] und [6]

C Parametersätze für Feuermuster

Feuermuster	$I_{gladapt}$	I_{fire}	I_{radapt}	V_{reset}	I_{stim}	sonstiges
Tonic Spiking	0	0	0	444	200	Neuron Nr. 3
	0	0	0	444	140	Neuron Nr. 5
Spike Frequency Adaptation	1000	300	500	444	360	Neuron Nr. 3
	1000	330	500	444	300	Neuron Nr. 5
Phasic Spiking	1000	330	500	444	223	Neuron Nr. 5
Initial Bursting	1023	440	500	290	330	Neuron Nr. 5
Phasic Bursting	1023	440	500	290	309	Neuron Nr. 5
Irregular Spiking	1023	390	500	294	340	Neuron Nr. 5

Tabelle 4: Bifurkationsparameter für verschiedene AdEx Feuermuster. Setup: visionlab2

Neuronparameter	Wert	Globaler Parameter	Wert
E_L	288	I_{breset}	1023
E_{syni}	0	I_{bstim}	1023
E_{synx}	572	int_op_bias	1023
I_{bexp}	819	V_{bexp}	1023
I_{convi}	1023	V_{bout}	1023
I_{convx}	1023	V_{br}	0
I_{gl}	121	V_{bstdf}	0
I_{intbbi}	614	V_{ccas}	800
I_{intbbx}	614	V_{clrc}	0
I_{pl}	20	V_{clra}	0
I_{rexp}	503	V_{dep}	0
$I_{spikeamp}$	820	V_{dllres}	200
V_{exp}	276	V_{dtc}	0
V_{syni}	0	V_{fac}	0
V_{syntci}	5	V_{gmax0}	1000
V_{syntcx}	811	V_{gmax1}	1000
V_{synx}	0	V_{gmax2}	1000
V_t	565	V_{gmax3}	1000
V_m	0	V_{stdf}	0
-	-	V_{thigh}	0

Tabelle 5: Skalierungsparameter, die für alle Feuermuster konstant gehalten wurden.

Literatur

[1] R. Brette & W. Gerstner: Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity, J Neurophysiol 94: 3637D3642, 2005

[2] W. Gerstner & W. Kistler: Spiking Neuron Models, Cambridge University Press, 2002

[3] M. Schwartz: Reproducing Biologically Realistic Regimes on a Highly-Accelerated Neuromorphic Hardware System, 2013, Universität Heidelberg

[4] A. Kononov: Testing of an Analog Neuromorphic Network Chip, 2011, Universität Heidelberg

[5] Scholarpedia, the peer-reviewed open-access encyclopedia: "Santiago Ramón y Cajal" `http://www.scholarpedia.org/article/Santiago_Ramon_y_Cajal`, 24.Juli 2013

[6] J.Schemmel, A.Grübl, S.Millner : Specification of the HICANN Microchip, Universiät Heidelberg, Juni 2013

[7] Dokumentation und Information zur Programmiersprache Python: `http://www.python.org/about`, 4.Juli 2013

[8] Offizielle Website des Qt Projekts: `http://www.qt-project.org`, 4.Juli 2013

[9] Offizielle Website der Pythonbibliothek matplotlib `http://www.matplotlib.org`, 4.Juli 2013

[10] BrainScaleS repository: `http://brainscales-r.kip-heidelberg.de`

[11] R.Naud, N. Marcille, C. Clopath, W. Gerstner: Firing patterns in the adaptive exponential integrate-and-fire model, Biological Cybernetics 99:335-347, 2008

[12] S.Millner: Develoment of a Multi-Compartment Neuron Model Emulation, 2012, Universität Heidelberg

[13] J. Touboul & R. Brette: Dynamics and bifurcations of the adaptive exponential integrate-and-fire model, Biological Cybernetics 99:319-334, 2008

[14] S. Millner et al.: A VLSI Implementation of the Adaptive Exponential Integrate-And-Fire Neuron Model, Advances in Neural Information Processing Systes, 23, 2010, p.1642-1650

[15] D.Brüderle et al.: A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems, Biological Cybernetics 104:263-296, 2011

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 08.Juli 2013,