Dissertation

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics

of the Ruperto-Carola University of Heidelberg, Germany

for the degree of
Doctor of Natural Sciences

presented by
Dipl.-Phys. Thomas Pfeil
born in Stuttgart, Germany

Date of oral examination: February 4, 2015

Exploring the potential of brain-inspired computing

Referees: Prof. Dr. Karlheinz Meier
Prof. Dr. Abigail Morrison

Declaration

- Publication I has been published as:

  Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., Schmuker, M., Brüderle, D., Schemmel, J., & Meier, K. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci. 7*(11).

- Publication II has been published as:

  Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., & Meier, K. (2012). Is a 4-bit synaptic weight resolution enough? - Constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci. 6*(90).

- Publication III has been published as:

  Pfeil, T., Scherzer, A.-C., Schemmel, J., & Meier, K. (2013). Neuromorphic learning towards nano second precision. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–5. IEEE Press.

- Publication IV has been published as:

  Schmuker, M., Pfeil, T., & Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proc. Natl. Acad. Sci. USA 111*(6), 2081–2086.

- Publication V has been submitted for publication in a peer-reviewed journal:

  Pfeil, T., Jordan, J., Tetzlaff, T., Grübl, A., Schemmel, J., Diesmann, M., & Meier, K. (2014). The effect of heterogeneity on decorrelation mechanisms in spiking neural networks: a neuromorphic-hardware study

Note that due to missing copyrights the re-print of the original publications is not allowed. Consequently, the arXiv.org (2014) versions of these publications are inserted into this thesis, respectively, including corrections and differing formatting than in the original publications. For Publication IV no arXiv.org (2014) version exists, and the camera-ready version sent to the journal was used.

Contents

Abstract

Exploring the potential of brain-inspired computing

The gap between brains and computers regarding both their cognitive capability and power efficiency is remarkably huge. Brains process information massively in parallel and its constituents are intrinsically self-organizing, while in digital computers the execution of instructions is deterministic and rather serial. The recent progress in the development of dedicated hardware systems implementing physical models of neurons and synapses enables to efficiently emulate spiking neural networks. In this work, we verify the design and explore the potential for brain-inspired computing of such an analog neuromorphic system, called *Spikey*. We demonstrate the versatility of this highly configurable substrate by the implementation of a rich repertoire of network models, including models for signal propagation and enhancement, general purpose classifiers, cortical models and decorrelating feedback systems. Network emulations on *Spikey* are highly accelerated and consume less than 1 nJ per synaptic transmission. The *Spikey* system, hence, outperforms modern desktop computers in terms of fast and efficient network simulations closing the gap to brains. During this thesis the stability, performance and user-friendliness of the *Spikey* system was improved integrating it into the neuroscientific tool chain and making it available for the community. The implementation of networks suitable to solve everyday tasks, like object or speech recognition, qualifies this technology to be an alternative to conventional computers. Considering the compactness, computational capability and power efficiency, neuromorphic systems may qualify as a valuable complement to classical computation.

Exploration des Potenzials von neuronaler Datenverarbeitung

Der Unterschied zwischen Gehirnen und Computern, sowohl in ihren kognitiven Fähigkeiten als auch in ihrer Energieeffizienz, ist bemerkenswert groß. Gehirne verarbeiten Informationen enorm parallel und ihre Bestandteile organisieren sich potenziell von selbst. Im Gegensatz dazu führen digitale Computer Programmbefehle in einer deterministischen und vielmehr sequenziellen Art und Weise aus. Der aktuelle Fortschritt in der Entwicklung spezieller Hardware, die Neuronen und Synapsen physikalisch nachbildet, ermöglicht effiziente Emulationen von neuronalen Netzen. In dieser Arbeit verifizieren wir den Aufbau von *Spikey*, einem hoch konfigurierbaren neuromorphen Chip. Zudem wurde das Potenzial dieses Systems in Bezug auf neuronale Datenverarbeitung untersucht. Wir zeigen die Vielseitigkeit von *Spikey* durch die Implementierung einer Vielfalt von Netzwerkmodellen. Diese beinhalten Modelle zur Signalweiterleitung und -aufbereitung, universelle Klassifizierer, kortikale Modelle und Dekorrelation in rückgekoppelten Systemen. Netzwerke werden auf *Spikey* beschleunigt emuliert und eine synaptische Übertragung benötigt weniger als 1 nJ an Energie. Somit lassen sich neuronale Netze schneller und stromsparender simulieren als mit Schreibtischcomputern, was uns der Energieeffizienz von Gehirnen näher bringt. Im Zuge dieser Arbeit wurden die Zuverlässigkeit, Geschwindigkeit und Benutzerfreundlichkeit des *Spikey*-Systems weiterentwickelt, so dass das System der Forschungsgemeinschaft zur Verfügung gestellt werden kann, und sich in die dort gebräuchliche Softwareumgebung integrieren lässt. Die Tatsache, dass wir auch Netzwerke aufsetzen konnten, die für alltägliche Aufgaben wie der Objekt- oder Spracherkennung verwendet werden können, macht diese Technologie zu einer Alternative zu herkömmlichen Computern. Bedenkt man die Kompaktheit, Rechenleistung und Energieeffizienz, könnten neuromorphe Systeme eine bahnbrechende Erweiterung zur klassischen Datenverarbeitung darstellen.

Chapter 1

Introduction

"The human brain performs computations inaccessible to the most powerful of today's computers – all while consuming no more power than a light bulb. Understanding how the brain computes reliably with unreliable elements, and how different elements of the brain communicate, can provide the key to a completely new category of hardware (Neuromorphic Computing Systems) and to a paradigm shift for computing as a whole. The economic and industrial impact is potentially enormous."

–

Human Brain Project (2014)

Artificial neural networks are composed of computation nodes, known as neurons, that are densely interconnected through contact points called synapses. The spiking activity of each neuron is a non-linear function of its inputs, and connections between neurons are plastic, and can be tuned by learning mechanisms (Section 1.1). Usually, emergent dynamics of network models are simulated on general purpose digital computers designed for serialized, centralized and deterministic data processing (von Neumann, 1993). In this work, we explore the potential of a neuromorphic computing substrate, in which each unit of biological neural networks, i.e., neurons and synapses, is represented by a physical implementation (Section 1.2). Consequently, computation takes place in parallel in all units and the separation between processing and memory vanishes, like in nervous systems.

Using analog electronics is one approach to implement such physical models of neural networks. As an example for this, the neuromorphic microchip *Spikey* comprises an analog VLSI (very-large-scale integration) implementation of a highly configurable neural network (Section 1.3). This computing substrate is portable, fast and energy-efficient, and computation is potentially massively parallel, plastic and robust. However, the design of algorithms for spiking neural networks, and especially for neuromorphic realizations of these, is subject to ongoing research. Here, we show the hardware implementation of a rich repertoire of state-of-the-art functional and non-functional network models (Section 1.4 and 1.5). These models range from plastic models for robust and precise signal processing over general purpose classifiers to models for fundamental research on correlations in recurrent systems.

The presented work does not only support the idea of the *Spikey* system to be a universal computing substrate, but also suggests the system to be ready as a tool for *Computational Neuroscience* and technical applications. Altogether, the system's versatility is shown by the enclosed publications, in which different networks are implemented on one and the same neuromorphic substrate. However, each of the five studies focuses on specific

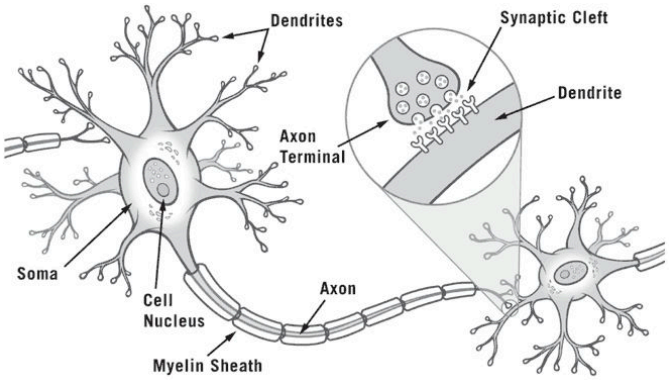


Figure 1.1: The structure of nerve cells (neurons) and neural networks. Spikes travel along axons and activate synapses (gap or contact between axon terminal and dendrite). Released neurotransmitters affect the membrane potential of the postsynaptic neuron. When the neuron receives sufficient activation, it fires a spike that travels along its axon to other neurons. Adapted from www.thethirdsource.org (2014).

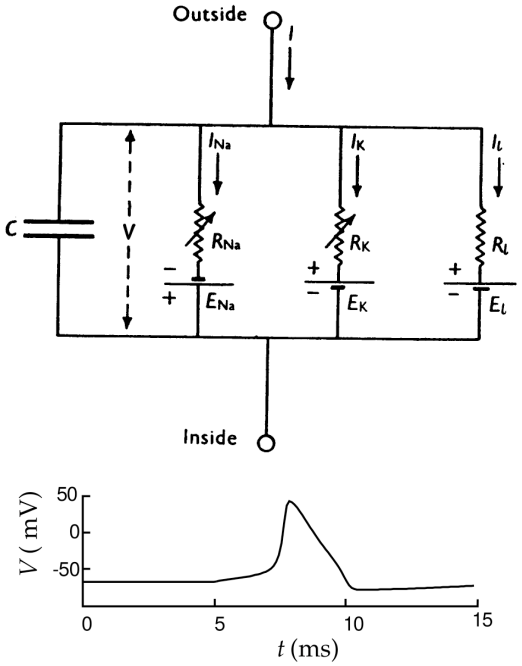
features of the system.

1.1 The nervous system

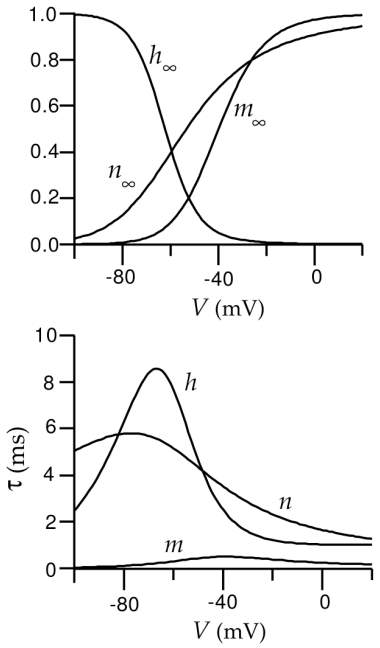
The nervous system, and in particular the neocortex of mammals that is thought to be responsible for higher functions, are composed of interconnected nerve cells (neurons). Neurons receive their input through synapses, at which action potentials (also known as spikes) arrive at the axonal terminal and trigger the release of neurotransmitters (Figure 1.1). These transmitters diffuse quickly to the dendrite of the postsynaptic cell and affect the neuron's membrane potential. Two types of synapses, namely excitatory and inhibitory synapses, either increase (depolarize) or decrease (hyperpolarize) the membrane potential, respectively. When sufficient excitatory spikes arrive and the membrane potential crosses a certain threshold, a spike is generated by the neuron itself and travels along its axon to other neurons. Without further stimulation the membrane potential relaxes to its ground state.

The above description simplifies a neuron to a point-like structure and exclusively considers biochemical synapses. Biological neurons, however, show more complex morphologies, electrophysiological properties and firing patterns (e.g., Connors & Gutnick, 1990; Gray & McCormick, 1996; Gibson et al., 1999; Nowak et al., 2003; Markram et al., 2004). For example, dendrites are observed to actively process signals (for a review, see London & Häusser, 2005) and neurons are shown to be electrically coupled (for a review, see Kumar & Gilula, 1996). Neuron models represent a tradeoff between the electrophysiological richness of detail and both mathematical tractability and computational complexity. For example, Hodgkin & Huxley (1952) formulated a model that describes the generation of action potentials in detail (see also Section 1.1.1). Multi-compartment models capture additional features like the spatial expansion of neurons (Dayan & Abbott, 2001) and the properties of active dendrites. In this work, the established leaky integrate-and-fire neuron model is used (for discussion see Section 3.1).

Rate neuron models represent a further abstraction of spiking neurons, and are usually defined by a single state variable that describes the neuron's average firing rate (Dayan & Abbott, 2001). Networks of such rate neurons perform well in many applications in machine



(a) Top: Electrical circuit representing the membrane. See text for details. Adapted from Hodgkin & Huxley (1952). Bottom: Membrane potential for current injection initiated at $t = 5\,\text{ms}$. Adapted from Dayan & Abbott (2001).



(b) Measured dependency of the gating variables n, m and h on the membrane potential V. For details see text and Equation 1.4. Adapted from Dayan & Abbott (2001).

Figure 1.2: The Hodgkin-Huxley neuron model.

learning (e.g., Bishop, 2006). However, there is evidence for the benefit of temporal codes, as generated by spiking neurons, for efficient information processing (see, e.g., review by Maass, 1997). Because spiking networks can process both temporal and rate codes, these networks are believed to be a suitable substrate for brain-inspired computation, and are used throughout this work.

1.1.1 The Hodgkin-Huxley neuron model

In the early 50s Hodgkin & Huxley (1952) proposed a mathematical description of the flow of electric current through the surface membrane of a giant nerve fibre. Amongst other observations, this model gives insight into the details of the generation of action potentials.

The membrane of the nerve cell is described by an electrical circuit with capacitance C (Figure 1.2(a)). Current passing the membrane is carried by ions flowing through resistances R that are in parallel with the capacity. Ionic currents are divided into a sodium (I_{Na}), potassium (I_{K}) and leakage current (I_{l}). The difference between the membrane potential and the equilibrium potentials, which is specific for each type of ion, determines the driving force for ionic currents (see voltage sources in Figure 1.2(a)). Thus, the derivative of the membrane potential can be written as

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = I + g_{\text{Na}}(E_{\text{Na}} - V) + g_{\text{K}}(E_{\text{K}} - V) + g_{\text{l}}(E_{\text{l}} - V), \tag{1.1}$$

where I is an externally applied current, e.g., through patch clamping techniques (e.g. Sakmann & Neher, 1984). While the conductance of the leakage g_{l} is assumed to be static, the conductances for sodium (g_{Na}) and potassium (g_{K}) are a function of membrane

potential and time. The probabilities of ions passing ion channels in the membrane are associated with gating variables n, m and h such that

$$g_{\mathrm{K}} = \overline{g_{\mathrm{K}}} n^4 \tag{1.2}$$

and

$$g_{\mathrm{Na}} = \overline{g_{\mathrm{Na}}} m^3 h. \tag{1.3}$$

where $\overline{g_{\mathrm{K}}}$ and $\overline{g_{\mathrm{Na}}}$ are constant. The derivative of each of these gating variables is modeled as

$$\tau_x(V)\frac{\mathrm{d}x}{\mathrm{d}t} = x_\infty(V) - x, \tag{1.4}$$

with $x \in \{n, m, h\}$. Note that both x_∞ and τ_x are dependent on the membrane potential (see Figure 1.2(b)).

The generation of an action potential is an interplay between sodium influx and delayed potassium outflow. If the cell is strongly depolarized, e.g., by an current injection or synaptic excitation, the open probability for sodium channels increases almost instantaneous and depolarization is amplified (see m in Figure 1.2(b)). Because h decreases with increasing membrane potential, this sodium influx slows down. Simultaneously, the open probability of potassium channels increases slowly and pulls the membrane back to its resting state.

The Hodgkin-Huxley neuron model has been shown to accurately reproduce membrane potentials of neurons (Hodgkin & Huxley, 1952), but is computationally rather expensive.

1.1.2 The leaky integrate-and-fire neuron model

The above Hodgkin-Huxley model can be simplified to the so-called leaky integrate-and-fire model (Dayan & Abbott, 2001), because the generation of action potentials is typically triggered when the membrane potential reaches a threshold voltage. Although the complex dynamics of voltage-dependent sodium and potassium channels is replaced by a simple threshold process, the leaky integrate-and-fire neuron model is a very useful description for neural activity. Its simplified dynamics allow for more efficient numerical simulations and easier mathematical tractability compared to the Hodgkin-Huxley model. Sub-threshold dynamics simplify to:

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = I + g_{\mathrm{l}}(E_{\mathrm{l}} - V). \tag{1.5}$$

When the membrane potential crosses the firing threshold

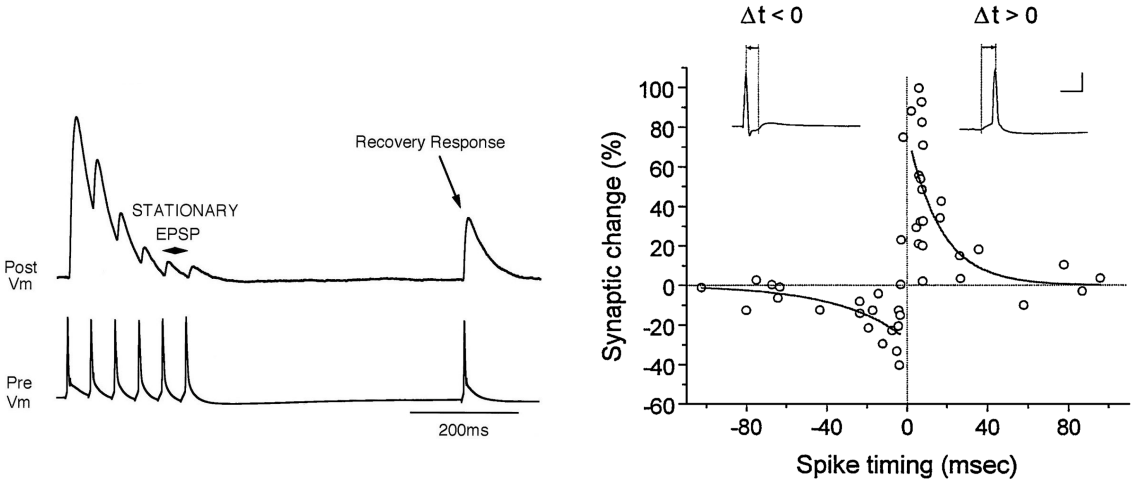
$$V > V_{\mathrm{th}} \tag{1.6}$$

the membrane potential is clamped to V_{reset} for the duration of the so-called refractory period τ_{ref}.

This neuron model is commonly used in neuroscience and is implemented on the neuromorphic system used in this study (for details, see Section 1.3). For an example trace, see membrane potential in Figure 5 of Publication V.

1.1.3 Synaptic transmissions

So far, we considered stimulations of neurons by externally applied currents I. In neural networks, neurons are interconnected by synapses and action potentials are mediated by synaptic transmissions. At arrival of action potentials at the presynaptic terminal the synapse emits transmitters that bind to the membrane of the postsynaptic neurons and



(a) The amplitude of excitatory postsynaptic potentials (EPSPs) decreases with frequent presynapic spikes until an equilibrium state is reached (stationary EPSPs). In the absence of further input the nerve cell recovers (recovery response). Adapted from Tsodyks & Markram (1997).

(b) Positive time intervals Δt between the pre- and postsynaptic spikes cause a positive change in the synaptic weight, and vice versa for $\Delta t < 0$. Here, exponential functions are fitted independently for positive and negative Δt. Adapted from Bi & Poo (2001) and based on data by Bi & Poo (1998).

Figure 1.3: Biological measurements of (a) short-term and (b) spike-timing dependent plasticity, respectively.

thus modifies the membrane's conductance (see Figure 1.1). This biochemical process can be modelled by

$$I = g_{\text{syn}}(t)(E_{\text{syn}} - V), \tag{1.7}$$

where E_{syn} is a static potential, also called reversal potential, and $g_{\text{syn}}(t)$ is the synaptic conductance course of the synapse. In this work, we use two types of synapses, $syn \in \{exc, inh\}$ for excitatory and inhibitory synapses, respectively, and synaptic conductances are modeled by exponentially decaying functions

$$\tau_{\text{syn}} \frac{\mathrm{d}g_{\text{syn}}}{\mathrm{d}t} = -g_{\text{syn}}, \tag{1.8}$$

where g_{syn} is step-wise increased for each incoming spike.

The computational benefit of other interactions between neurons than these biochemical transmissions, e.g., gap junctions (Kumar & Gilula, 1996), is believed to be secondary, and is hence not considered in the following.

1.1.4 Short-term plasticity

Tsodyks & Markram (1997), among others, found that synaptic efficacy changes with presynaptic activity within a range of several hundred milliseconds, which is called short-term plasticity (STP). Two types of STP have been observed in experiments, short-term depression (STD; see Figure 1.3(a)) and short-term facilitation (STF). The former is caused by depletion of neurotransmitters, while the latter can be attributed to the influx of calcium at the axon terminal, which, in turn, increases the release probability of neurotransmitters. The distribution of neurotransmitter resources can be described by the following equations (Tsodyks & Markram, 1997):

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \frac{I}{\tau_{\mathrm{rec}}}, \tag{1.9}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\frac{E}{\tau_{\mathrm{inact}}} + U_{\mathrm{SE}} R \delta(t - t_{\mathrm{AP}}), \tag{1.10}$$

$$I = 1 - R - E, \tag{1.11}$$

where R, I, and E are the recovered, inactive and effective fraction of resources, respectively. Each presynaptic spike at time t_{AP} instantaneously activates resources U_{SE} from the recovered state R. The recovered and effective partitions rise and relax with the time constants τ_{rec} and τ_{inact}, respectively. The synaptic efficacy is proportional to the effective partition E.

STP is believed to not only be an unavoidable consequence of synaptic physiology, but may also be responsible for the processing of temporal information on time scales relevant in daily life, e.g., in motor control or speech recognition (Tsodyks & Wu, 2013).

1.1.5 Spike-timing dependent plasticity

The strength of connections, the so-called synaptic weight, between neurons can be modulated by plasticity mechanisms. Hebb (1949) has postulated: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." Summarized and simplified this means: "what fires together wires together".

Almost half a century later Bi & Poo (1998), among others, have observed the dependence of the change of synaptic weights on the precise timing of action potentials, namely spike-timing dependent plasticity (STDP; Figure 1.3(b); see also Morrison et al., 2008; Sjöström & Gerstner, 2010). If the presynaptic neuron fires before the postsynaptic one, the synaptic weight increases, and vice versa. This set of experimental data suggests separate exponential fits for pre- before postsynaptic and post- before presynaptic spike pairings:

$$x(\Delta t) = \exp\left(-\frac{|\Delta t|}{\tau_{\mathrm{STDP}}}\right), \tag{1.12}$$

with Δt the time interval between a pre- and postsynaptic spike and τ_{STDP} the STDP time constant.

Weight changes Δw may not only depend on the spike-timing dependence $x(\Delta t)$, but are usually scaled by a factor $F(w)$ that depends on the current weight w (for references and examples, see Publication II and Morrison et al., 2007):

$$\Delta w = F(w) x(\Delta t). \tag{1.13}$$

Although spike correlations are considered in the order of several ten milliseconds, long-term plasticity affects synaptic weights on a time scale of seconds to years (Sjöström & Gerstner, 2010). STDP is believed to underly learning and memory formation in brains (e.g., Bi & Poo, 2001; Sjöström et al., 2008), and stabilizes network activity by modifying the weight of either excitatory (Kempter et al., 2001) or inhibitory synapses (Froemke et al., 2007; Vogels et al., 2011).

Note that other biological recordings revealed a large variety of fit functions $x(\Delta t)$ (for a review, see Abbott & Nelson, 2000), and more complex measurement protocols (e.g., Sjöström et al., 2001) suggested more complex plasticity models (e.g., Pfister & Gerstner, 2006; Clopath et al., 2010; Graupner & Brunel, 2012), not further addressed in this work.

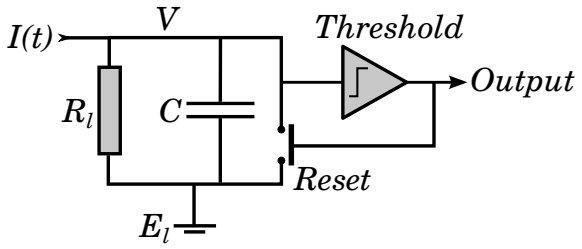


Figure 1.4: Schematized electronical model of a neuron. The capacity C of the cell membrane is charged by an external current I and discharged by the leak resistance R_l. If the external current is strong enough, the firing threshold is crossed, the neuron is reset and the output spike is propagated to other neurons within the network. Adapted from Brüderle (2014).

1.2 Physical models of spiking neural networks

To study brain function in terms of its information processing properties, recently, in the research field of *Computational Neuroscience* many biologically realistic network models have been developed. The behavior of the constituents of these models, namely neurons and synapses, is mostly formulated by differential equations. However, the complex interaction of spiking neurons, and hence coupling of these equations, requires methods to solve these equations. To this end, most researches apply numerical methods and simulate their networks with software running on computers based on the von Neumann (1993) architecture. For simplicity, (modified) von Neumann (1993) architectures and parallel arrangements of these, like graphics processing units (GPUs), are summarized as *conventional* computers, in the following. Network simulators utilize conventional computers and allow the investigation of network dynamics of several ten thousand complex multi-compartment (Hines & Carnevale, 1997; Hines et al., 2008) or of more than a billion point neurons (Gewaltig & Diesmann, 2007; Kunkel et al., 2014). In simulations, each variable within the networks can be observed, which allows for profound analyses of complete network dynamics on multiple spatial and temporal scales. This is in contrast to the sparse data that can be acquired from brains, although recording techniques improved significantly within the last decades (for reviews, see Helmchen, 2005; Dodt et al., 2007; Weckstrom, 2010; Gross, 2011). However, simulations of spiking neural networks are time and power consuming, impeding their efficient mass application to solve real world problems.

The gap in efficiency between conventional computers and brains is caused by the fundamentally different ways data is processed. While in conventional computers instructions are executed sequentially, neurons process data massively in parallel. Parallel arrangements of computers in clusters of these bridge the gap in terms of parallel computation, but are far away from the power efficiency of brains (for discussion see Section 3.4).

An alternative approach to overcome these limitations of conventional computer are systems that are composed of physical models of neurons and synapses. An intuitive illustration of this approach is a mechanical implementation of a leaky integrate-and-fire model (Stefanini, 2012). A leaky water bucket mounted rotatably about its horizontal axis represents a neuron and is filled with water waves (spikes) released by two other buckets. Only if the water waves arrive coincidently, this bucket is temporarily full enough to flip and release water itself.

This concept can be transferred to VLSI systems where neurons and synapses are electronic models of biological neurons (see reviews by Renaud et al., 2010; Indiveri et al., 2011; Hasler & Marr, 2013). Electronic components show similar properties compared

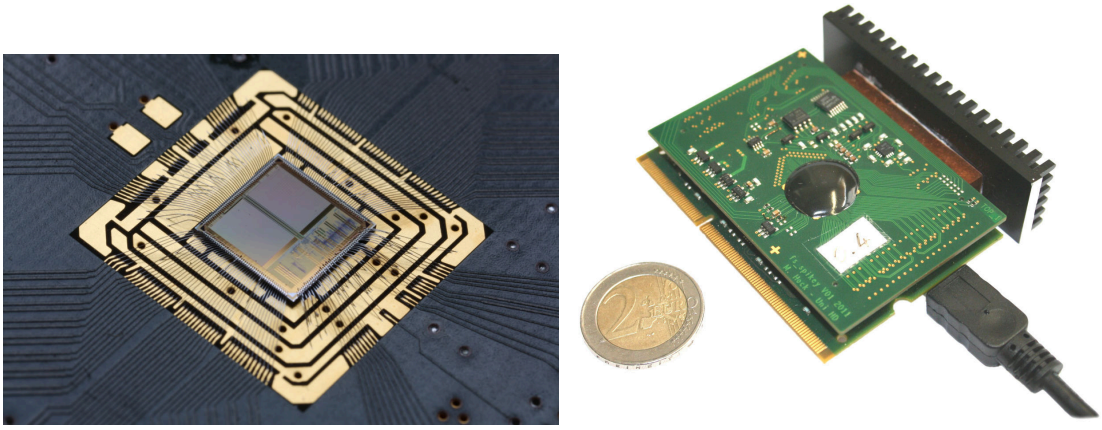
to the biochemical structures in nervous tissue and can be manufactured with very high density using submicron complementary metal-oxide-semiconductor (CMOS) technology. For example, capacitances resemble the insulating membrane of nerve cells and transistors resemble populations of ion channels, respectively (see Figure 1.4 and Mead, 1990). To emphasize the difference between simulations on conventional computers and simulations on analog neuromorphic hardware, we use the term *emulations* for the latter.

Such analog VLSI systems can be categorized in two classes, real-time and accelerated systems, which does not exclude intermediate implementations. Real-time systems are usually optimized for applications in prosthetics and robotics (e.g., Floreano & Mattiussi, 2001; Indiveri, 2001; Serrano-Gotarredona et al., 2009; Arthur et al., 2012), and usually consume little power, e.g., by operating transistors in the weak inversion regime (Badoni et al., 2006; Indiveri et al., 2006; Hafliger, 2007; Vogelstein et al., 2007; Indiveri et al., 2009; Giulioni et al., 2008; Serrano-Gotarredona et al., 2009; Renaud et al., 2010; Yu & Cauwenberghs, 2010; Brink et al., 2013; Benjamin et al., 2014). Furthermore, these real-time systems can be used for spike-based communication to brain tissue (e.g., Bontorin et al., 2007) or neuromorphic sensors (e.g., silicon retinas as reviewed by Posch et al. (2014)).

In contrast, operating the circuits outside the weak inversion regime allows for highly accelerated systems, which means that the ratio between the time constants of the hardware network and their biological counterparts is very small. The *Spikey* system used in this work (see Section 1.3 and Publication I) and the system developed by Wijekoon & Dudek (2008, 2012) show similar design goals. In particular, both systems comprise STP and STDP circuits, and are accelerated by a factor of 10^4 and 10^3, respectively, compared to biological real-time. However, although single neurons and synapses have been analyzed for the latter, we do not know of any network emulations larger than three neurons for this system. Recently, highly accelerated photonic systems have been developed, e.g., by Hurtado et al. (2012) and Coomans et al. (2011), in which lasers (neurons) are used to trigger light beams (spikes) that travel along optical fibres to excite other lasers (for a review, see Shastri et al., 2014; Tait et al., 2014). Time scales in these photonic systems are up to 10^9 times smaller than in biological systems (Hurtado et al., 2012).

The acceleration of analog neuromorphic systems can potentially be preserved while scaling to larger network sizes, because all neurons intrinsically run in parallel. However, scaling becomes more difficult with higher acceleration. Neuromorphic chips in real-time systems usually communicate to each other via conventional signalling techniques, often using the address-event representation (AER) protocol (as, e.g., used by Serrano-Gotarredona et al., 2006). For accelerated systems these techniques quickly reach their limit in terms of data bandwidth between the chips (Jeltsch, 2010). One approach to break this limit is the wafer-scale integration of neuromorphic chips improving the data bandwidth drastically (Schemmel et al., 2010; Brüderle et al., 2011). The interconnection of multiple of these wafers allows for networks consisting of more than one million neurons. To exploit this multi-wafer system, the modularity and connectivity of biological networks has to be considered (Perin et al., 2011). Connections between nearby neurons within a modular structure are dense and should be realized within a wafer, while connections between distant neurons or modules are sparse and may be realized across wafers in order to save the rather limited bandwidth between wafers.

An expanded definition of neuromorphic systems also includes purely digital systems. The approach by Merolla et al. (2011, 2014) comprises dedicated electronic circuits for each neuron within the network, like in analog systems, but neurons are implemented fully digital and signals are integrated numerically. Another system uses conventional



(a) Microphotograph of the *Spikey* chip.

(b) The *Spikey* neuromorphic system

Figure 1.5: The *Spikey* chip is mounted on a printed circuit board, which is smaller than a cigarette pack. This board, in turn, is plugged into another board hosting the FPGA, memory, and many other features (see main text).

integration techniques on off-the-shelf low power processors that are highly interconnected compared to conventional supercomputers (Furber et al., 2014).

Neuromorphic systems are usually compact and computation is energy efficient, fast and robust. This makes them perfect candidates for computationally intensive real-world applications and attract economical interest. Besides heavy long-term funding by the European Union (SenseMaker, 2005; FACETS, 2010; BrainScaleS, 2014; Human Brain Project, 2014), the United States of America (SyNAPSE, 2014; The Brain Initiative, 2014) and other governments (Brain/MINDS, 2014; Government of Canada, 2014; Brainnetome, 2014), recently private companies initiated the development of neuromorphic systems, e.g., IBM (Merolla et al., 2014; IBM, 2014), Qualcomm (2013), Intel (Roy et al., 2014), ARM (Furber et al., 2014), HP (HP, 2010; Jo et al., 2010) and Brain Corporation (2014). However, efficient neural algorithms for spiking neural networks are still rare.

1.3 The *Spikey* neuromorphic system

The core component of the used neuromorphic substrate is the very-large-scale integration (VLSI) chip called *Spikey* (Figure 1.5). It comprises two blocks of 192 neurons receiving input from 256 synapses each, totaling 98304 synapses. Within the neural network information is processed exclusively with analog circuits, except the digital storage of parameters, e.g., synaptic weights.

The system is optimized for acceleration of the hardware network. The actual acceleration factor depends on the time constants of the implemented network model. In the current setup the membrane time constant τ of hardware neurons can be configured approximately between 0.2 µs and 1.5 µs (Scherzer, 2013). Consequently, for neocortical neurons ($\tau \approx 10$ ms has been reported by McCormick et al., 1985) the acceleration is approximately 10^4-fold. In other words, the acceleration factor is not a fixed property of the hardware system, but specifies the translation from hardware to biological parameter domain, e.g., here, 10 s of biological network time correspond to 1 ms on hardware. Throughout this work, we define the acceleration to be 10^4-fold, except for the network model shown in Publication III, in which the time constants of the biological model are much smaller, and

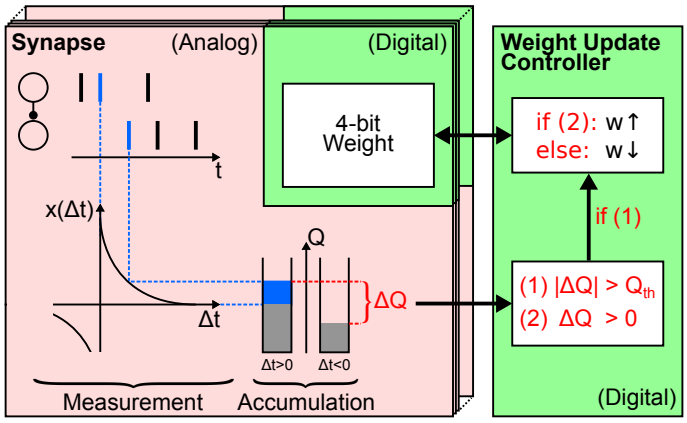


Figure 1.6: STDP on the *Spikey* chip. Time intervals Δt are measured between pairs of pre- and postsynaptic spikes. These measurements are accumulated on capacitors for $\Delta t > 0$ and < 0, respectively. A global update controller sequentially reads out the charge on these capacitors and updates to synaptic weight according to the evaluation of both charges: If condition (1) is fulfilled, a weight update will be triggered. Otherwise, the synapse continues to measure. In case of an update, condition (2) determines two alternative and freely configurable rules to update the weight. In this work, we use an additive update rule (see Publication II).

hence the acceleration factor is reduced.

Besides its high acceleration, the network on the *Spikey* chip is highly configurable (for details, see Publication I). To improve the comparability to existing literature, neuron and synapse parameters are specified in biological parameter domain. During configuration, these parameters are translated to hardware domain considering the acceleration factor and the fact that voltages on the *Spikey* chip differ from those observed in biological tissue. However, not all parameters can be configured individually for each neuron and synapse. Besides stimulation through on-chip recurrent connections, data packages containing the time and source address of spikes can be fed into the hardware network. During experiment execution the spiking activity of all neurons and the membrane potential of one neuron are recordable. For a details about the specification of the *Spikey* chip and the data flow, see Publication I.

Two plasticity mechanisms are implemented on the *Spikey* chip, short- and long-term plasticity. Short-term plasticity is implemented for each axon (Schemmel et al., 2007) and modulates the maximum conductance in dependence on the presynaptic activity (for details, see Section 1.1.4). In contrast to a possible mixture of facilitation and depression in the model of Tsodyks & Markram (1997), on hardware synapses are limited to either facilitating or depressing STP. This mechanism is helpful to stabilize the activity of recurrent networks of excitatory neurons, which mutually reinforce their activity without short-term plasticity (Brüderle et al., 2010; Bill et al., 2010, and both balanced random network and cortical model in Publication I).

The hardware implementation of pair-wise STDP is split into local circuits in each synapse and a global controller for each block of synapses (see Section 1.1.5 and Schemmel et al., 2006, respectively). In each synapse, the time intervals between pre- and postsynaptic, and parallely between post- and presynaptic spikes are measured and accumulated on two separate capacitors. Rows of synapses are processed sequentially by a global controller that reads out these accumulated measurements and the 4-bit synaptic weights.

In dependence on the evaluation of the charge on both capacitors the synaptic weight is changed accordingly (see Figure 1.6). The rather small resolution of synaptic weights and the analog implementation of the measurement and accumulation circuits for STDP keeps synapses compact. This allows for a high density and hence a large number of synapses, which takes the typically high synapse count for each neuron into account (e.g., in cortical circuits, several thousand synapses per neuron have been reported in the collection by Potjans & Diesmann, 2014). The electronics to update the digital synaptic weight is area consuming and hence shared between synapses (see Publication II). However, this design has limitations compared to the STDP model described in Section 1.1.5 and its implementation in software. Firstly, the resolution of synaptic weights is limited on hardware compared to the floating point precision usually applied in software (for the discussion of two-valued synapse models, see Publication II). Secondly, in most STDP models the occurance of single spike pairs is resolved (see, e.g., review by Morrison et al., 2008), while on hardware multiple spike pairs may be accumulated due to the limited frequency with which synapses are evaluated. This may be not of concern, because measurement protocols of STDP are usually composed of several to tens of spike pairs (e.g., Cassenaer & Laurent, 2007, 2012), and so far there is no consensus that a single spike pair actually induces a weight change. The effect of the above limitations of hardware STDP, among others, is studied on an exemplary network in Publication II.

Due to fluctuations in the manufacturing process all analog circuits suffer from device variations. In a parallel arrangement of analog circuits, i.e., physical models of neurons and synapses, device variations manifest in a spatially disordered pattern, called *fixed-pattern noise*. However, these variations of parameters across neurons and synapses are determined by the time of production. Then, they are approximately constant over time and can be calibrated for. In contrast, the hardware network is subject to *temporal noise* including electronic noise and transient experiment conditions, e.g., fluctuations of the chip temperature. Reproducibility is a prerequisite for systematic investigations of physical experiments in general and neural networks in particular. The ratio between signal and noise has to be significant to test hypothesis. Consequently, network dynamics are generally averaged across multiple trials. In Publication V both fixed-pattern and temporal noise is quantified and discussed.

During this thesis, Schemmel (2014), Grübl (2014), Hock (2014b) and Hartel (2014), among others, developed a new hardware infrastructure to operate the *Spikey* chip. This system is connected via universal serial bus (USB) 2.0 to the host computer, and is equipped with a modern FPGA to manage the data flow and on-board memory to buffer data (for details, see Publication I). Thereby, the communication bandwidth could be increased approximately by one order of magnitude compared to the older backplane system (for a detailed description of the backplane system, see Brüderle, 2009). For all studies we used chips of identical version 4, but the studies in Publication II to III were performed on the backplane system, while the studies in Publication IV and V used the USB device. Because both the backplane and USB system implement the same functionality as seen from the host computer and the *Spikey* chip, the system description in Publication I is also valid for the USB system. However, due to the on-board analog-to-digital converter (ADC), an oscilloscope to observe the membrane potential becomes obsolete. Additionally, the USB system provides a gyro and acceleration sensor, a wireless chip, general purpose digital inputs/outputs (I/Os) and serial high speed data links, e.g., to interconnect multiple of these systems. During the scope of this thesis, this new system was put into operation, which included assembly, testing and software development. Obstacles observed during this process lead to improved revision of the *Spikey* chip and the board carrying the chip.

The development of the *Spikey* neuromorphic hardware system started in the year 2005. It was designed as a highly configurable mixed-signal system to prototype network models on neuromorphic hardware. The neural network is mostly implemented with analog circuits, and the communication to the host computer for configuration, simulation and observation is mostly implemented with digital signalling techniques. The digital infrastructure is described by Grübl (2007) in detail. The layouts of the analog neuron and synapses circuits are published in Schemmel et al. (2006, 2007) and Indiveri et al. (2011), respectively. Brüderle (2009) and his students (Kaplan, 2008; Müller, 2008; Bill, 2008; Friedmann, 2009; Schilling, 2010; Vogginger, 2010; Jeltsch, 2010; Müller, 2011; Pfeil, 2011; Petkov, 2012)[1] established the basis for a standardized operation of the system including tests and troubleshooting. The access to the system was encapsulated by several layers of abstraction (for details, see Brüderle, 2009; Brüderle et al., 2009, and Publication I). The top layer is an interface for the standardized network description language `PyNN`, which allows users to model and run neural networks without any detailed knowledge of the hardware system. In addition, methods were developed to calibrate the initially heterogeneous hardware network (fixed-pattern noise) and the behavior of analog circuits was characterized (Brüderle, 2009). In the course of these measurements the chip was revised three times up to version four that is used throughout the work at hand. The easy operation of the hardware system via `PyNN` and calibrations facilitated the implementation of several simple network models (for listing, see Publication I). At this stage STDP was measured isolatedly for single synapses (Schemmel et al., 2006; Müller, 2008; Brüderle, 2009), but was not stable and not used autonomously for multiple synapses.

1.4 Publication overview

The publications underlying the thesis at hand (see Chapter 2) are briefly summarized in the next paragraph, and are discussed thematically in the subsequent Section.

The first study emphasizes the high configurability of the hardware system by successfully implementing six network models on the neuromorphic *Spikey* system (Publication I). Secondly, and thirdly, the hardware implementation of STDP is investigated (Publication II) and for the first time demonstrated in functional networks on accelerated hardware (Publication III). Fourthly, the successful implementation of a classifier network proposes the utilization of the *Spikey* system to solve real-world computing tasks (Publication IV). Lastly, the hardware system is used as a stand-alone tool to show that heterogeneity in the computing substrate increases correlations in recurrent neural networks (Publication V).

1.5 Brain-inspired computing on a neuromorphic substrate

Recently, novel neuromorphic systems inspired by the architecture of brains overcame the limitation of the word-at-a-time thinking of conventional computers by processing data massively in parallel, and by breaking the separation of memory and computation. Potential advantages of this design are compactness, scalability, power efficiency, plasticity and robustness. Programs are formulated as descriptions of spiking neural networks and are not necessarily deterministic, but may learn and evolve during runtime. Compared to the rich repertoire of brain-inspired machine learning algorithms mostly based on networks consisting of rate neurons (e.g., Bishop, 2006), algorithms for spiking neural networks are rare.

[1] Only diploma thesis are listed, bachelor thesis are not included.

In this work, we explore the potential of brain-inspired computing with spiking neural networks in general and of the *Spikey* system in particular: The versatility of the highly programmable *Spikey* chip is shown by the implementation of many different network models (Section 1.5.1), the capability of learning was analyzed (Section 1.5.2), novel algorithms for real world application were developed (Section 1.5.3) and network dynamics of interest for the neuroscientific community were investigated (Section 1.5.4).

1.5.1 Universality of the computing substrate

Central processing units (CPUs) based on the von Neumann (1993) architecture are Turing complete. This means that these processors can simulate the logic of any computer algorithm, if we ignore limitations of finite memory. CPUs are programmed by a sequence of stored instructions, hence the same computing substrate can be used for many different applications.

Another example for a highly re-configurable and versatile system is an field-programmable gate array (FPGA), which comprises integrated circuits composed of logic gates and random-access memory (RAM) blocks. The interconnection of logic gates can be programmed, and hence FPGAs can be used to prototype application-specific integrated circuits (ASICs), in which the logic is hardwired. Of course, FPGAs usually contain fewer building blocks, are slower and consume more power than ASICs, but the uncomplicated and rapid re-configuration of FPGAs without the need to manufacture new prototype chips makes them a perfect candidate whenever fast and cheap prototyping is needed in digital chip design.

Our objective is to transfer this concept of a re-configurable computing substrate to neuromorphic systems. Obviously, configurable neurons and their interconnections, namely configurable and optionally plastic synapses, are candidates for basic building blocks. The *Spikey* neuromorphic system is one of the first prototypes of such a system that additionally provides a high acceleration of computation, and is programmed by connecting neurons and parametrizing neurons and synapses. Configurability of this sort allows for rapid prototyping of neural algorithms, in other words models of spiking neural networks, compared to the otherwise slow and expensive development of new application-specific neuromorphic systems. Once an algorithm is fully developed, still an application-specific chip can be derived from the configurable system. Then, the integration density and power efficiency may be further improved. However, efficient algorithms for real world tasks are rare and subject to recent research. Note that Maass (1996) has shown Turing completeness for spiking neural networks, and hence any algorithm can potentially be implemented with ideal and sufficiently large networks.

For fast prototyping of neural algorithms, not only the configurability of the computing substrate is crucial, but also the ease to configure it. Therefore, the standardized model description language `PyNN` is used to interface the system. On the one hand, `PyNN` simplifies the daily usage of the system by both encapsulating and abstracting network descriptions and embedding the hardware into the neuroscientific tool chain, on the other hand `PyNN` facilitates the usage of the system by external users (e.g., see Publication IV).

Although the neuron and synapse models integrated on the *Spikey* chip are quite simple (Section 1.1.2), a large number of network models could already be implemented (Publication I to V). The configurability of long-term plasticity on hardware was investigated separately (Publication II).

The actual system incorporates neuron, synapse and plasticity models that were state-of-the-art at the time of the chip development in 2005. In the meanwhile recording methods for nervous tissue improved, which led to a deeper theoretical understanding of brain func-

tion (for selected recording techniques, see Helmchen, 2005; Weckstrom, 2010; Gross, 2011; Ahrens et al., 2013). Thus, the extraction of key features promoting efficient computation stays a challenging task. This is especially true during the design phase of highly configurable, and potentially universal, neuromorphic substrates, because revising them is time-consuming and expensive.

1.5.2 Plasticity and learning in hardware networks

The capability of neural networks to learn a task and adapt to a changing environment is believed to be mostly realized by the modification and formation of connections between neurons. Additionally, plasticity plays a key role in network formation, maintenance and repair after lesions. In the context of the vast parameter space of possible connections between neurons on the *Spikey* chip, the programming of these connections could potentially be replaced by learning them. Consequently, the size of configuration data is reduced drastically, and the acceleration of the substrate causes learning to be likely faster than the configuration of all connections. Furthermore, plasticity mechanisms may enhance the robustness of hardware networks against a changing environment (see Section 3.5).

One approach for autonomous learning in spiking neural networks is spike-timing dependent plasticity (STDP; Section 1.1.5), an asymmetric form of Hebbian learning, which is widely used in network models (e.g., Gerstner et al., 1996; Song et al., 2000; Izhikevich, 2007; Cassenaer & Laurent, 2007, 2012). STDP is a local plasticity rule inherent to synapses, and depends only on pre- and postsynaptic activity, in an unsupervised fashion. Due to the large number of synapses, the potentially spatial separation between pre- and postsynaptic activity (in terms of computing cores and memory), and the typically long duration of simulations, STDP is computationally expensive on conventional computers (Morrison et al., 2007; Davies et al., 2012). On the *Spikey* chip, emulations are not slowed down by the activation of plasticity, because a plasticity rule that approximates STDP is implemented with analog circuits massively in parallel in each synapse. Limited by the available chip area, the STDP implementation on hardware is a tradeoff between the complexity of the STDP model (see Section 1.1.5 and Publication II) and the resources needed for each synapse (Schemmel et al., 2006). On the *Spikey* chip the instantaneous acquisition of temporal correlations between pre- and postsynaptic activity locally in each synapse is separated from a comparatively infrequent and global evaluation of these correlations. Depending on this evaluation the 4-bit synaptic weight that is also stored locally in each synapse is updated. The functionality of this design is validated by means of an example network that performs synchrony detection (Publication II). This study is of preparative nature with regard to the design of the wafer-scale system that comprises circuits for STDP in each synapse very similar to these in the *Spikey* chip, but allows for a more configurable evaluation. In addition, we demonstrate the computational power of massively parallel plasticity in each synapse by means of implementing a network model that is inspired by the auditory system of owls (Publication III). Plasticity mechanisms in this networks autonomously converts noisy spike input to a precise phase-locked signal that is the basis for accurate sound localization.

In contrast to the above on-chip learning, synaptic weights on the *Spikey* chip can also be learned by using the *chip in the loop* (see Publication IV and the liquid state machine in Publication I). This means that a network is emulated on the chip and dependent on its activity the connectivity on the chip is re-configured. Thus, plasticity is computed off-chip, e.g., by the host computer, which is effective for computationally simple plasticity models, e.g., models depending on the neuron's firing rates. Computationally complex plasticity models, like STDP, do not benefit from this approach, because off-chip solutions lack the

massive parallelism exploited by on-chip STDP.

In addition to long-term plasticity, the *Spikey* chip also implements short-term plasticity (Section 1.1.4; Schemmel et al., 2007), which is used to stabilize network activity in several of the presented networks (see balanced random network and cortical model in Publication I).

1.5.3 Hardware usage for practical applications

For everyday applications neuromorphic systems offer several advantages compared to conventional processors. Considering the computational power in terms of running neural networks, neuromorphic systems are usually compact and power efficient making them perfect candidates for mobile systems. However, dependent on the computational task, e.g., deterministic operations like summations, conventional processors may be still advantageous. Hence, neuromorphic devices can be seen as complementary to conventional computers. Due to the fact that both computing architectures are usually implemented in silicon, the advantages of both architectures can be combined by efficiently integrating them on a single chip. One example of such an integration is a general purpose processor that is used to control plasticity on the wafer-scale system (Friedmann et al., 2013; Friedmann, 2013).

Although not integrated on a single chip, in this work, we show a hybrid implementation of a classifier for multivariate data (Publication IV). Real-world data is pre-processed on the host computer, and then filtered and classified by a network implemented on the *Spikey* chip. Furthermore, similar filters may be used to effectively reduce the information in large data streams to the components that are of interest for further data processing. This approach is inspired by the nervous system, of which we know that it performs this task with remarkable efficiency, e.g., in the context of vision. However, the *Spikey* system is still of prototype nature, its network size is rather limited, and additional features of neural networks may be missing to address complex real-world problems (for discussion, see Chapter 3).

The application of accelerated systems in robotics is difficult, because the time constants of sensor data and the computing substrate are likely to differ by several orders of magnitude. If sensor data is encoded by spikes, the sensor activity has to be buffered and temporally compressed such that its temporal structure is compatible with the accelerated hardware. On first sight this sounds unsuitable, but because the network time of the accelerated system is in general much shorter than the acquisition time of the sensor, multiple network emulations can be executed during acquisition. This allows to improve the network performance by presenting several variants of the sensor data (e.g., Cireşan et al., 2012) or to virtually increase the network size by sequentially emulating partitions of the network. However, these partitions have to be arranged in a feed-forward structure to avoid unsolvable interdependencies. In the case of real-valued sensor data, very small response times can be achieved with accelerated hardware (Publication IV).

1.5.4 Relevance of the hardware for neuroscience

Besides practical applications, neuromorphic systems have a great potential for *Computational Neuroscience*. Especially accelerated systems can provide high performance computing substrates allowing for studies not feasible for conventional supercomputers, e.g., for studies in which short emulation times are of advantage (see last paragraph in this Section and Müller, 2014). In addition, network emulations on *Spikey* are usually by several orders of magnitude more power efficient than simulations of the same network model on conventional computers (see Section 3.4). Note that the number of neurons on the *Spikey*

chip is rather small and limits its spectrum of application in the neuroscience community. However, a larger system operating at comparable acceleration is under development (Schemmel et al., 2010; Brüderle et al., 2011).

Programs for neuromorphic hardware systems, i.e., network descriptions, are formulated in a language that is native for the hardware they are executed on, because biological entities are mapped to their physical representations on hardware. However, so far, few spiking network models exist that are suitable for a neuromorphic implementation. Usually, model parameters are homogeneous and fine-tuned, and there is no standardized procedure to test these models for robustness against parameter variations that are inevitable for analog neuromorphic hardware. Consequently, following the design of the nervous system, e.g., the recovery after a stroke, networks and plasticity rules have to be developed to improve and stabilize network performance on unreliable computing substrates. Population coding and redundancy is one approach towards robust functional networks on analog neuromorphic hardware (see Publication IV and cortical model in Publication I). In combination with online plasticity, noisy and heterogeneous system can even become precise measuring instruments (Publication III).

While in Publication I and III established network models were transferred to the *Spikey* system and the focus lies on the system's versatility, the study in Publication V revealed that the decorrelating effect of recurrent inhibition is diminished by heterogeneity in network parameters, and hence demonstrates exemplarily how neuroscientific questions can be approached by analog neuromorphic hardware. The results of the latter study are not only relevant for the understanding of dynamics in rather non-functional networks, but are of general importance for the implementation of brain-inspired networks on heterogeneous substrates, e.g., analog neuromorphic hardware.

Here, we exploited the high acceleration of the system to gather a vast amount of data providing proper statistics for our analysis. Despite parallelization in conventional computers, the *Spikey* chip outperforms desktop computers in terms of fast and power-efficient network simulations (see Section 1 in Supplements of Publication V and Section 3.4). Of course, comparable network simulations can be parallelized across multiple CPU cores or CPUs, however, this does not significantly affect the power efficiency of computation. Furthermore, accelerated operation is of even more advantage if networks can not be simulated in parallel. This will for example be the case, if simulations depend on previous simulations like in iterative processes (e.g., see iterative learning in Publication IV).

Chapter 2

Publications

I Six networks on a universal neuromorphic computing substrate

Thomas Pfeil[1,†], Andreas Grübl[1,†], Sebastian Jeltsch[1,†], Eric Müller[1,†], Paul Müller[1,†], Mihai A. Petrovici[1,†], Michael Schmuker[2,3,†], Daniel Brüderle[1], Johannes Schemmel[1], and Karlheinz Meier[1]

[1] *Kirchhoff-Institute for Physics, Universität Heidelberg, Heidelberg, Germany*
[2] *Neuroinformatics & Theoretical Neuroscience, Freie Universität Berlin, Berlin, Germany*
[3] *Bernstein Center for Computational Neuroscience Berlin, Berlin, Germany*

Abstract

In this study, we present a highly configurable neuromorphic computing substrate and use it for emulating several types of neural networks. At the heart of this system lies a mixed-signal chip, with analog implementations of neurons and synapses and digital transmission of action potentials. Major advantages of this emulation device, which has been explicitly designed as a universal neural network emulator, are its inherent parallelism and high acceleration factor compared to conventional computers. Its configurability allows the realization of almost arbitrary network topologies and the use of widely varied neuronal and synaptic parameters. Fixed-pattern noise inherent to analog circuitry is reduced by calibration routines. An integrated development environment allows neuroscientists to operate the device without any prior knowledge of neuromorphic circuit design. As a showcase for the capabilities of the system, we describe the successful emulation of six different neural networks which cover a broad spectrum of both structure and functionality.

[†] These authors contributed equally to this work. See acknowledgements for details.

1 Introduction

By nature, computational neuroscience has a high demand for powerful and efficient devices for simulating neural network models. In contrast to conventional general-purpose machines based on a von-Neumann architecture, neuromorphic systems are, in a rather broad definition, a class of devices which implement particular features of biological neural networks in their physical circuit layout (Mead, 1989; Indiveri et al., 2009; Renaud et al., 2010). In order to discern more easily between computational substrates, the term *emulation* is generally used when referring to neural networks running on a neuromorphic back-end.

Several aspects motivate the neuromorphic approach. The arguably most characteristic feature of neuromorphic devices is inherent parallelism enabled by the fact that individual neural network components (essentially neurons and synapses) are physically implemented *in silico*. Due to this parallelism, scaling of emulated network models does not imply slowdown, as is usually the case for conventional machines. The hard upper bound in network size (given by the number of available components on the neuromorphic device) can be broken by scaling of the devices themselves, e.g., by wafer-scale integration (Schemmel et al., 2010) or massively interconnected chips (Merolla et al., 2011). Emulations can be further accelerated by scaling down time constants compared to biology, which is enabled by deep submicron technology (Schemmel et al., 2006, 2010; Brüderle et al., 2011). Unlike high-throughput computing with accelerated systems, real-time systems are often specialized for low power operation (e.g., Indiveri et al., 2006; Farquhar & Hasler, 2005).

However, in contrast to the unlimited model flexibility offered by conventional simulation, the network topology and parameter space of neuromorphic systems are often dedicated for predefined applications and therefore rather restricted (e.g., Serrano-Gotarredona et al., 2006; Merolla & Boahen, 2006; Akay, 2007; Chicca et al., 2007). Enlarging the configuration space always comes at the cost of hardware resources by occupying additional chip area. Consequently, the maximum network size is reduced, or the configurability of one aspect is decreased by increasing the configurability of another. Still, configurability costs can be counterbalanced by decreasing precision. This could concern the size of integration time steps (Imam et al., 2012a), the granularity of particular parameters (Pfeil et al., 2012) or fixed-pattern noise affecting various network components. At least the latter can be, to some extent, moderated through elaborate calibration methods (Neftci & Indiveri, 2010; Brüderle et al., 2011; Gao et al., 2012).

In this study, we present a user-friendly integrated development environment that can serve as a universal neuromorphic substrate for emulating different types of neural networks. Apart from almost arbitrary network topologies, this system provides a vast configuration space for neuron and synapse parameters (Schemmel et al., 2006; Brüderle et al., 2011). Reconfiguration is achieved on-chip and does not require additional support hardware. While some models can easily be transferred from software simulations to the neuromorphic substrate, others need modifications. These modifications take into account the limited hardware resources and compensate for fixed-pattern noise (Kaplan et al., 2009; Brüderle et al., 2009, 2010, 2011; Bill et al., 2010). In the following, we show six more networks emulated on our hardware system, each requiring its own hardware configuration in terms of network topology and neuronal as well as synaptic parameters.

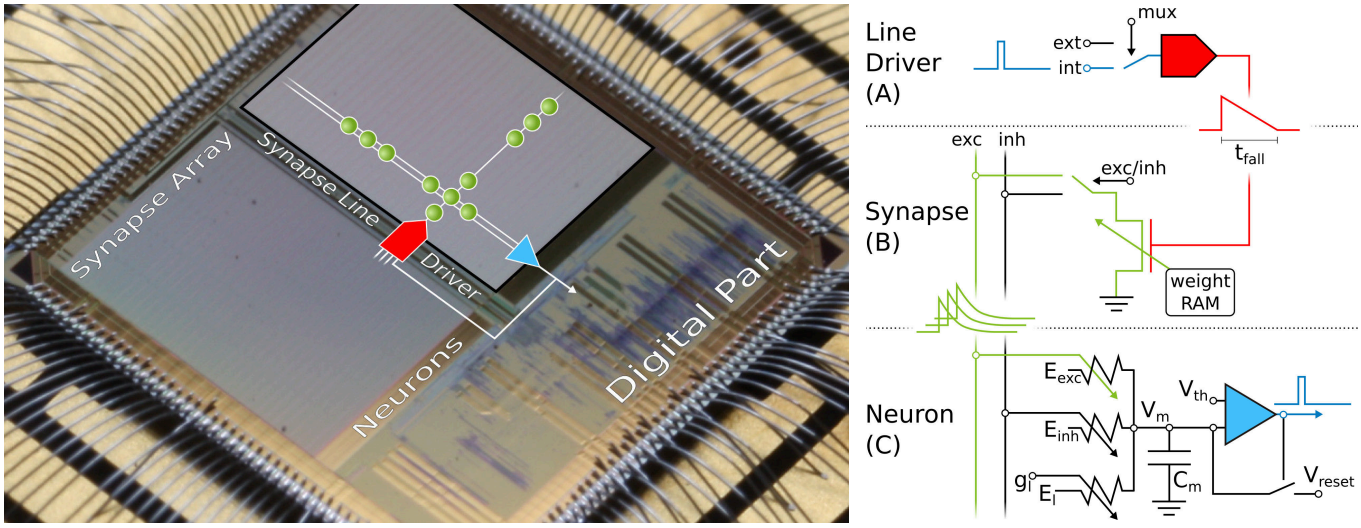


Figure 1: Microphotograph of the *Spikey* chip (fabricated in a 180 nm CMOS process with die size $5 \times 5\,\mathrm{mm}^2$). Each of its 384 neurons can be arbitrarily connected to any other neuron. In the following, we give a short overview of the technical implementation of neural networks on the *Spikey* chip. **(A)** Within the synapse array 256 synapse line drivers convert incoming digital spikes (blue) into a linear voltage ramp (red) with a falling slew rate t_{fall}. For simplicity, the slew rate of the rising edge is not illustrated here, because it is chosen very small for all emulations in this study. Each of these synapse line drivers are individually driven by either another on-chip neuron (int), e.g., the one depicted in (C), or an external spike source (ext). **(B)** Within the synapse, depending on its individually configurable weight w_i, the linear voltage ramp (red) is then translated into a current pulse (green) with exponential decay. These postsynaptic pulses are sent to the neuron via the excitatory (exc) and inhibitory (inh) input line, shared by all synapses in that array column. **(C)** Upon reaching the neuron circuit, the total current on both input lines is converted into conductances, respectively. If the membrane potential V_{m} crosses the firing threshold V_{th}, a digital pulse (blue) is generated, which can be recorded and fed back into the synapse array. After any spike, V_{m} is set to V_{reset} for a refractory time period of τ_{refrac}. Neuron and synapse line driver parameters can be configured as summarized in Table 1.

2 The Neuromorphic System

The central component of our neuromorphic hardware system is the neuromorphic microchip *Spikey*. It contains analog very-large-scale integration (VLSI) circuits modeling the electrical behavior of neurons and synapses (Figure 1). In such a *physical model*, measurable quantities in the neuromorphic circuitry have corresponding biological equivalents. For example, the membrane potential V_{m} of a neuron is modeled by the voltage over a capacitor C_{m} that, in turn, can be seen as a model of the capacitance of the cell membrane. In contrast to numerical approaches, dynamics of physical quantities like V_{m} evolve continuously in time. We designed our hardware systems to have time constants approximately 10^4 times faster than their biological counterparts allowing for high-throughput computing. This is achieved by reducing the size and hence the time constant of electrical components, which also allows for more neurons and synapses on a single chip. To avoid confusion between hardware and biological domains of time, voltages and currents, all parameters are specified in biological domains throughout this study.

2.1 The Neuromorphic Chip

On *Spikey* (Figure 1), a VLSI version of the standard leaky integrate-and-fire (LIF) neuron model with conductance-based synapses is implemented (Dayan & Abbott, 2001):

$$C_\mathrm{m}\frac{\mathrm{d}V_\mathrm{m}}{\mathrm{d}t} = -g_\mathrm{l}(V_\mathrm{m} - E_\mathrm{l}) - \sum_i g_i(V_\mathrm{m} - E_i) \tag{1}$$

For its hardware implementation see Figure 1, Schemmel et al. (2006) and Indiveri et al. (2011).

Synaptic conductances g_i (with the index i running over all synapses) drive the membrane potential V_m towards the reversal potential E_i, with $E_i \in \{E_\mathrm{exc}, E_\mathrm{inh}\}$. The time course of the synaptic activation is modeled by

$$g_i(t) = p_i(t) \cdot w_i \cdot g_i^\mathrm{max} \tag{2}$$

where g_i^max are the maximum conductances and w_i the weights for each synapse, respectively. The time course $p_i(t)$ of synaptic conductances is a linear transformation of the current pulses shown in Figure 1 (green), and hence an exponentially decaying function of time. The generation of conductances at the neuron side is described in detail by Indiveri et al. (2011), postsynaptic potentials are measured by Schemmel et al. (2007).

The implementation of spike-timing dependent plasticity (STDP; Bi & Poo, 1998; Song et al., 2000) modulating w_i over time is described in Schemmel et al. (2006) and Pfeil et al. (2012). Correlation measurement between pre- and post-synaptic action potentials is carried out in each synapse, and the 4-bit weight is updated by an on-chip controller located in the digital part of the *Spikey* chip. However, STDP will not be further discussed in this study.

Short-term plasticity (STP) modulates g_i^max (Schemmel et al., 2007) similar to the model by Tsodyks & Markram (1997) and Markram et al. (1998). On hardware, STP can be configured individually for each synapse line driver that corresponds to an axonal connection in biological terms. It can either be facilitating or depressing.

The propagation of spikes within the *Spikey* chip is illustrated in Figure 1 and described in detail by Schemmel et al. (2006). *Spikes* enter the chip as time-stamped events using standard digital signaling techniques that facilitate long-range communication, e.g., to the host computer or other chips. Such digital packets are processed in discrete time in the digital part of the chip, where they are transformed into digital *pulses* entering the synapse line driver (blue in Figure 1A). These pulses propagate in continuous time between on-chip neurons, and are optionally transformed back into digital spike packets for off-chip communication.

2.2 System Environment

The *Spikey* chip is mounted on a network module described and schematized in Fieres et al. (2004) and Figure 2, respectively. Digital spike and configuration data is transferred via direct connections between a field-programmable gate array (FPGA) and the *Spikey* chip. Onboard digital-to-analog converter (DAC) and analog-to-digital converter (ADC) components supply external parameter voltages to the *Spikey* chip and digitize selected voltages generated by the chip for calibration purposes. Furthermore, up to eight selected membrane voltages can be recorded in parallel by an oscilloscope. Because communication between a host computer and the FPGA has a limited bandwidth that does not satisfy real-time operation requirements of the *Spikey* chip, experiment execution is controlled by

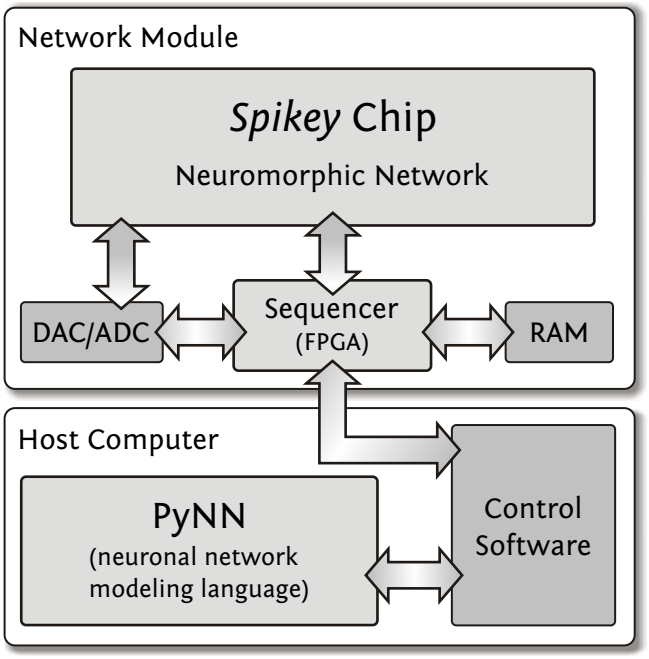


Figure 2: Integrated development environment: User access to the *Spikey* chip is provided using the `PyNN` neural network modeling language. The control software controls and interacts with the network module which is operating the *Spikey* chip. The RAM size (512 MB) limits the total number of spikes for stimulus and spike recordings to approx. $2 \cdot 10^8$ spikes. The required data for a full configuration of the *Spikey* chip has a size of approx. 100 kB.

the FPGA while operating the *Spikey* chip in continuous time. To this end, all experiment data is stored in the local random access memory (RAM) of the network module. Once the experiment data is transferred to the local RAM, emulations run with an acceleration factor of 10^4 compared to biological real-time. This acceleration factor applies to all emulations shown in this study, independent of the size of networks.

Execution of an experiment is split up into three steps (Figure 2). First, the *control software* within the memory of the host computer generates configuration data (Table 1, e.g., synaptic weights, network connectivity, etc.), as well as input stimuli to the network. All data is stored as a sequence of commands and is transferred to the memory on the network module. In the second step, a playback sequencer in the FPGA logic interprets this data and sends it to the *Spikey* chip, as well as triggers the emulation. Data produced by the chip, e.g., neuronal activity in terms of spike times, is recorded in parallel. In the third and final step, this recorded data stored in the memory on the network module is retrieved and transmitted to the host computer, where they are processed by the control software.

Having a control software that abstracts hardware greatly simplifies modeling on the neuromorphic hardware system. However, modelers are already struggling with multiple incompatible interfaces to software simulators. That is why our neuromorphic hardware system supports `PyNN`, a widely used application programming interface (API) that strives for a coherent user interface, allowing portability of neural network models between different software simulation frameworks (e.g., `NEST` or `NEURON`) and hardware systems (e.g., the *Spikey* system). For details see Gewaltig & Diesmann (2007); Eppler et al. (2009) for `NEST`, Carnevale & Hines (2006); Hines et al. (2009) for `NEURON`, Brüderle et al. (2011, 2009) for

the *Spikey* chip, and Davison et al. (2009, 2010) for `PyNN`, respectively.

2.3 Configurability

In order to facilitate the emulation of network models inspired by biological neural structures, it is essential to support the implementation of different (cortical) neuron types. From a mathematical perspective, this can be achieved by varying the appropriate parameters of the implemented neuron model (Equation 1).

To this end, the *Spikey* chip provides 2969 different analog parameters (Table 1) stored on current memory cells that are continuously refreshed from a digital on-chip memory. Most of these cells deliver individual parameters for each neuron (or synapse line driver), e.g., leakage conductances g_l. Due to the size of the current-voltage conversion circuitry it was not possible to provide individual voltage parameters, such as, e.g., E_l, E_exc and E_inh, for each neuron. As a consequence, groups of 96 neurons share most of these voltage parameters. Parameters that can not be controlled individually are delivered by global current memory cells.

In addition to the possibility of controlling analog parameters, the *Spikey* chip also offers an almost arbitrary configurability of the network topology. As illustrated in Figure 1, the fully configurable *synapse array* allows connections from synapse line drivers (located alongside the array) to arbitrary neurons (located below the array) via synapses whose weights can be set individually with a 4-bit resolution. This limits the maximum fan-in to 256 synapses per neuron, which can be composed of up to 192 synapses from on-chip neurons, and up to 256 synapses from external spike sources. Because the total number of neurons exceeds the number of inputs per neuron, an all-to-all connectivity is not possible. For all networks presented in this study, the connection density is much lower than realizable on the chip, which supports the chosen trade-off between inputs per neuron and total neuron count.

2.4 Calibration

Device mismatch that arises from hardware production variability causes fixed-pattern noise, which causes parameters to vary from neuron to neuron as well as from synapse to synapse. Electronic noise (including thermal noise) also affects dynamic variables, as, e.g., the membrane potential V_m. Consequently, experiments will exhibit some amount of both neuron-to-neuron and trial-to-trial variability given the same input stimulus. It is, however, important to note that these types of variations are not unlike the neuron diversity and response stochasticity found in biology (Gupta et al., 2000; Maass et al., 2002; Marder & Goaillard, 2006; Rolls & Deco, 2010).

To facilitate modeling and provide repeatability of experiments on arbitrary *Spikey* chips, it is essential to minimize these effects by calibration routines. Many calibrations have directly corresponding biological model parameters, e.g., membrane time constants (described in the following), firing thresholds, synaptic efficacies or PSP shapes. Others have no equivalents, like compensations for shared parameters or workarounds of defects (e.g., Kaplan et al., 2009; Bill et al., 2010; Pfeil et al., 2012). In general, calibration results are used to improve the mapping between biological input parameters and the corresponding target hardware voltages and currents, as well as to determine the dynamic range of all model parameters (e.g., Brüderle et al., 2009).

While the calibration of most parameters is rather technical, but straightforward (e.g., all neuron voltage parameters), some require more elaborate techniques. These include the calibration of τ_m, STP as well as synapse line drivers, as we describe later for individual

Scope	Name	Type	Description
Neuron circuits (A)	n/a	$\mathsf{i_n}$	Two digital configuration bits activating the neuron and readout of its membrane voltage
	g_l	$\mathsf{i_n}$	Bias current for neuron leakage circuit
	τ_refrac	$\mathsf{i_n}$	Bias current controlling neuron refractory time
	E_l	$\mathsf{s_n}$	Leakage reversal potential
	E_inh	$\mathsf{s_n}$	Inhibitory reversal potential
	E_exc	$\mathsf{s_n}$	Excitatory reversal potential
	V_th	$\mathsf{s_n}$	Firing threshold voltage
	V_reset	$\mathsf{s_n}$	Reset potential
Synapse line drivers (B)	n/a	$\mathsf{i_l}$	Two digital configuration bits selecting input of line driver
	n/a	$\mathsf{i_l}$	Two digital configuration bits setting line excitatory or inhibitory
	$t_\mathrm{rise}, t_\mathrm{fall}$	$\mathsf{i_l}$	Two bias currents for rising and falling slew rate of presynaptic voltage ramp
	g_i^max	$\mathsf{i_l}$	Bias current controlling maximum voltage of presynaptic voltage ramp
Synapses (B)	w	$\mathsf{i_s}$	4-bit weight of each individual synapse
STP related (C)	n/a	$\mathsf{i_l}$	Two digital configuration bits selecting short-term depression or facilitation
	U_SE	$\mathsf{i_l}$	Two digital configuration bits tuning synaptic efficacy for STP
	n/a	$\mathsf{s_l}$	Bias voltage controlling spike driver pulse length
	τ_rec, τ_facil	$\mathsf{s_l}$	Voltage controlling STP time constant
	I	$\mathsf{s_l}$	Short-term facilitation reference voltage
	R	$\mathsf{s_l}$	Short-term capacitor high potential
STDP related (D)	n/a	$\mathsf{i_l}$	Bias current controlling delay for presynaptic correlation pulse (for calibration purposes)
	$A_{+/-}$	$\mathsf{s_l}$	Two voltages dimensioning charge accumulation per (anti-)causal correlation measurement
	n/a	$\mathsf{s_l}$	Two threshold voltages for detection of relevant (anti-)causal correlation
	τ_STDP	g	Voltage controlling STDP time constants

Table 1: List of analog current and voltage parameters as well as digital configuration bits. Each with corresponding model parameter names, excluding technical parameters that are only relevant for correctly biasing analog support circuitry or controlling digital chip functionality. Electronic parameters that have no direct translation to model parameters are denoted *n/a*. The membrane capacitance is fixed and identical for all neuron circuits ($C_\mathrm{m} = 0.2\,\mathrm{nF}$ in biological value domain). Parameter types: (i) controllable for each corresponding circuit: 192 for neuron circuits (denoted with subscript n), 256 for synapse line drivers (denoted with subscript l), 49152 for synapses (denoted with subscript s), (s) two values, shared for all even/odd neuron circuits or synapse line drivers, respectively, (g) global, one value for all corresponding circuits on the chip. All numbers refer to circuits associated to one synapse array and are doubled for the whole chip. For technical reasons, the current revision of the chip only allows usage of one synapse array of the chip. Therefore, all experiments presented in this paper are limited to a maximum of 192 neurons. For parameters denoted by (A) see Equation 1 and Schemmel et al. (2006), for (B) see Figure 1, Equation 2 and Dayan & Abbott (2001), for (C) see Schemmel et al. (2007) and for (D) see Schemmel et al. (2006) and Pfeil et al. (2012).

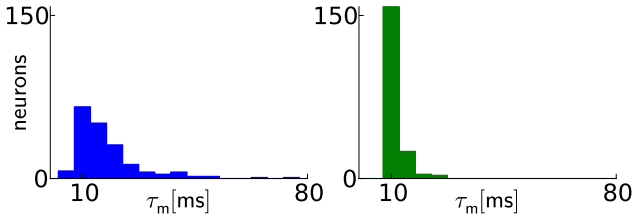


Figure 3: Calibration results for membrane time constants: Before calibration (left), the distribution of τ_m values has a median of $\widetilde{\tau_\mathrm{m}} = 15.1\,\mathrm{ms}$ with 20th and 80th percentiles of $\tau_\mathrm{m}^{20} = 10.3\,\mathrm{ms}$ and $\tau_\mathrm{m}^{80} = 22.1\,\mathrm{ms}$, respectively. After calibration (right), the distribution median lies closer to the target value and narrows significantly: $\widetilde{\tau_\mathrm{m}} = 11.2\,\mathrm{ms}$ with $\tau_\mathrm{m}^{20} = 10.6\,\mathrm{ms}$ and $\tau_\mathrm{m}^{80} = 12.0\,\mathrm{ms}$. Two neurons were discarded, because the automated calibration algorithm did not converge.

network models. The membrane time constant $\tau_\mathrm{m} = C_\mathrm{m}/g_\mathrm{l}$ differs from neuron to neuron mostly due to variations in the leakage conductance g_l. However, g_l is independently adjustable for every neuron. Because this conductance is not directly measurable, an indirect calibration method is employed. To this end, the threshold potential is set below the resting potential. Following each spike, the membrane potential is clamped to V_reset for an absolute refractory time τ_refrac, after which it evolves exponentially towards the resting potential E_l until the threshold voltage triggers a spike and the next cycle begins. If the threshold voltage is set to $V_\mathrm{th} = E_\mathrm{l} - 1/e \cdot (E_\mathrm{l} - V_\mathrm{reset})$, the spike frequency equals $1/(\tau_\mathrm{m}+\tau_\mathrm{refrac})$, thereby allowing an indirect measurement and calibration of g_l and therefore τ_m. For a given τ_m and $\tau_\mathrm{refrac} = const$, V_th can be calculated. An iterative method is applied to find the best-matching V_th, because the exact hardware values for E_l, V_reset and V_th are only known after the measurement. The effect of calibration on a typical chip can best be exemplified for a typical target value of $\tau_\mathrm{m} = 10\,\mathrm{ms}$. Figure 3 depicts the distribution of τ_m of a typical chip before and after calibration.

The STP hardware parameters have no direct translation to model equivalents. In fact, the implemented transconductance amplifier tends to easily saturate within the available hardware parameter ranges. These non-linear saturation effects can be hard to handle in an automated fashion on an individual circuit basis. Consequently, the translation of these parameters is based on STP courses averaged over several circuits.

3 Hardware Emulation of Neural Networks

In the following, we present six neural network models that have been emulated on the *Spikey* chip. Most of the emulation results are compared to those obtained by software simulations in order to verify the network functionality and performance. For all these simulations the tool `NEST` (Gewaltig & Diesmann, 2007) or `NEURON` (Carnevale & Hines, 2006) is used.

3.1 Synfire Chain with Feedforward Inhibition

Architectures with a feedforward connectivity have been employed extensively as computational components and as models for the study of neuronal dynamics. Synfire chains are feedforward networks consisting of several neuron groups where each neuron in a group projects to neurons in the succeeding group.

They have been originally proposed to account for the presence of behaviorally-related, highly precise firing patterns (Baker et al., 2001; Prut et al., 1998). Further properties of such structures have been studied extensively, including activity transport (Aertsen et al., 1996; Diesmann et al., 1999; Litvak et al., 2003), external control of information flow (Kremkow et al., 2010), computational capabilities (Abeles et al., 2004; Vogels & Abbott, 2005; Schrader et al., 2010), complex dynamic behavior (Yazdanbakhsh et al., 2002) and their embedding into surrounding networks (Aviel et al., 2003; Tetzlaff et al., 2005; Schrader et al., 2008). Kremkow et al. (2010) have shown that feedforward inhibition can increase the selectivity to the initial stimulus and that the local delay of inhibition can modify this selectivity.

3.1.1 Network Topology

The presented network model is an adaptation of the *feedforward network* described in Kremkow et al. (2010).

The network consists of several neuron groups, each comprising $n_{\mathrm{RS}} = 100$ excitatory regular spiking (RS) and $n_{\mathrm{FS}} = 25$ inhibitory fast spiking (FS) cells. All neurons are modeled as LIF neurons with exponentially decaying synaptic conductance courses. According to Kremkow et al. (2010) all neurons have identical parameters.

As shown in Figure 4A, RS neurons project to both RS and FS populations in the subsequent group while the FS population projects to the RS population in its local group. Each neuron receives a fixed number of randomly chosen inputs from each presynaptic population. The first group is stimulated by a population of n_{RS} external spike sources with identical connection probabilities as used for RS groups within the chain.

Two different criteria are employed to assess the functionality of the emulated synfire chain. The first, straightforward benchmark is the stability of signal propagation. An initial synchronous stimulus is expected to cause a stable propagation of activity, with each neuron in an RS population spiking exactly once. Deviations from the original network parameters can cause the activity to grow rapidly, i.e., each population emits more spikes than its predecessor, or stall pulse propagation.

The second, broader characterization follows Kremkow et al. (2010), who has analyzed the response of the network to various stimuli. The stimulus is parametrized by the variables a and σ. For each neuron in the stimulus population a spike times are generated by sampling them from a Gaussian distribution with common mean and standard deviation. σ is defined as the standard deviation of the spike times of all source neurons. Spiking activity that is evoked in the subsequent RS populations is characterized analogously by measuring a and σ.

Figure 4C shows the result of a software simulation of the original network. The filter properties of the network are reflected by a separatrix dividing the state space shown in Figure 4C and D into two areas, each with a different fixed point. First, the basin of attraction (dominated by red circles in Figure 4C) from which stable propagation can be evoked and second, the remaining region (dominated by crosses in Figure 4C) where any initial activity becomes extinguished. This separatrix determines which types of initial input lead to a stable signal propagation.

3.1.2 Hardware Emulation

The original network model could not be mapped directly to the *Spikey* chip because it requires 125 neurons per group, while on the chip only 192 neuron circuits are available. Further constraints were caused by the fixed synaptic delays, which are determined by the

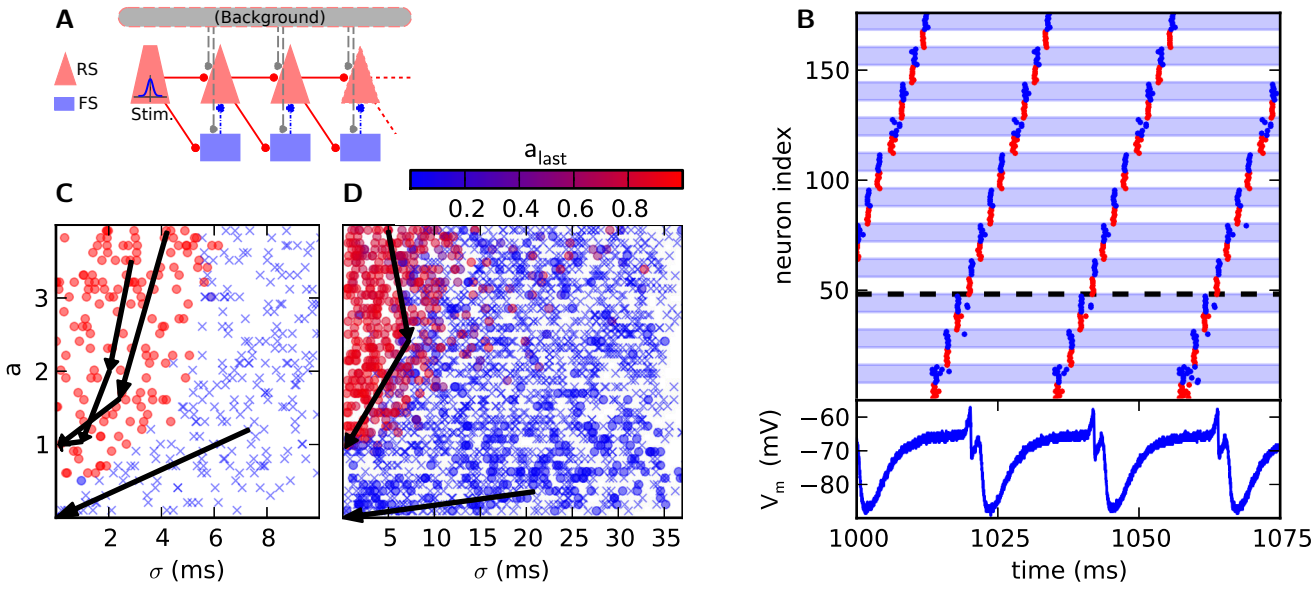


Figure 4: **(A)** Synfire chain with feedforward inhibition. The background is only utilized in the original model, where it is implemented as random Gaussian current. For the presented hardware implementation it has been omitted due to network size constraints. As compensation for missing background stimuli, the resting potential was increased to ensure a comparable excitability of the neurons. **(B)** Hardware emulation. Top: Raster plot of pulse packet propagation 1000 ms after initial stimulus. Spikes from RS groups are shown in red and spikes from FS groups are denoted by blue color and background. Bottom: Membrane potential of the first neuron in the fourth RS group, which is denoted by a dashed horizontal line. The cycle duration is approximately 20 ms. **(C)** State space generated with software simulations of the original model. The position of each marker indicates the (σ, a) parameters of the stimulus while the color encodes the activity in the RS population of the third synfire group. Lack of activity is indicated with a cross. The evolution of the pulse packet parameters is shown for three selected cases by a series of arrows. Activity either stably propagates with fixed point $(\sigma, a) = (0.1\,\text{ms}, 1)$ or extinguishes with fixed point $(\sigma, a) = (0\,\text{ms}, 0)$. **(D)** Same as (C), but emulated on the FACETS chip-based system. The activity in the last group is located either near $(\sigma, a) = (0\,\text{ms}, 0)$ or $(0.3\,\text{ms}, 1)$. The difference to software simulations is explained in Section 3.1.2.

speed of signal propagation on the chip. The magnitude of the delay is approximately 1 ms in biological time.

By simple modifications of the network, we were able to qualitatively reproduce both benchmarks defined in Section 3.1.1. Two different network configurations were used, each adjusted to the requirements of one benchmark. In the following, we describe these differences, as well as the results for each benchmark.

To demonstrate a stable propagation of pulses, a large number of consecutive group activations was needed. The chain was configured as a loop by connecting the last group to the first, allowing the observation of more pulse packet propagations than there are groups in the network.

The time between two passes of the pulse packet at the same synfire group needs to be maximized to allow the neurons to recover (see voltage trace in Figure 4B). This is accomplished by increasing the group count and consequently reducing the group size.

As too small populations cause an unreliable signal propagation, which is mainly caused by inhomogeneities in the neuron behavior, $n_{\mathrm{RS}} = n_{\mathrm{FS}} = 8$ was chosen as a satisfactory trade-off between propagation stability and group size. Likewise, the proportion of FS neurons in a group was increased to maintain a reliable inhibition. To further improve propagation properties, the membrane time constant was lowered for all neurons by raising g_{l} to its maximum value. The strength of inhibition was increased by setting the inhibitory synaptic weight to its maximum value and lowering the inhibitory reversal potential to its minimum value. Finally, the synaptic weights $RS_{\mathrm{i}} \rightarrow RS_{\mathrm{i}+1}$ and $RS_{\mathrm{i}} \rightarrow FS_{\mathrm{i}+1}$ were adjusted. With these improvements we could observe persisting synfire propagation on the oscilloscope 2 h wall-clock time after stimulation. This corresponds to more than 2 years in biological real-time.

The second network demonstrates the filtering properties of a hardware-emulated synfire chain with feedforward inhibition. This use case required larger synfire groups than in the first case as otherwise, the total excitatory conductance caused by a pulse packet with large σ was usually not smooth enough due to the low number of spikes. Thus, three groups were placed on a single chip with $n_{\mathrm{RS}} = 45$ and $n_{\mathrm{FS}} = 18$. The resulting evolution of pulse packets is shown in Figure 4D. After passing three groups, most runs resulted in either very low activity in the last group or were located near the point $(0.3\,\mathrm{ms}, 1)$, as illustrated in Figure 4D.

Emulations on hardware differ from software simulations in two important points: First, the separation in the parameter space of the initial stimulus is not as sharply bounded, which is demonstrated by the fact that occasionally, significant activity in the last group can be evoked by stimuli with large σ and large a, as seen in Figure 4D. This is a combined effect due to the reduced population sizes and the fixed pattern noise in the neuronal and synaptic circuits. Second, a stimulus with a small a can evoke weak activity in the last group, which is attributed to a differing balance between excitation and inhibition. In hardware, a weak stimulus causes both, the RS and FS populations to response weakly which leads to a weak inhibition of the RS population, allowing the pulse to reach the last synfire group. Hence, the pulse fades slowly instead of being extinguished completely. In the original model, the FS population is more responsive and prevents the propagation more efficiently.

Nevertheless, the filtering properties of the network are apparent. The quality of the filter could be improved by employing the original group size, which would require using a large-scale neuromorphic device (see, e.g., Schemmel et al., 2010).

Our hardware implementation of the synfire chain model demonstrates the possibility to run extremely long lasting experiments due to the high acceleration factor of the hardware system. Because the synfire chain model itself does not require sustained external stimulus, it could be employed as an autonomous source of periodic input to other experiments.

3.2 Balanced Random Network

Brunel (2000) reports *balanced random networks* (BRNs) exhibiting, among others, asynchronous irregular network states with stationary global activity.

3.2.1 Network Topology

BRNs consist of an inhibitory and excitatory population of neurons, both receiving feedforward connections from two populations of Poisson processes mimicking background activity. Both neuron populations are recurrently connected including connections within the populations. All connections are realized with random and sparse connections of prob-

ability p. In this study, synaptic weights for inhibitory connections are chosen four times larger than those for excitatory ones. In contrast to the original implementation using 12500 neurons, we scaled this network by a factor of 100 while preserving its firing behavior.

If single cells fire irregularly, the *coefficient of variation*

$$C_{\mathrm{V}} = \frac{\sigma_{\mathrm{T}}}{\overline{T}} \tag{3}$$

of interspike intervals has values close to or higher than one (Dayan & Abbott, 2001). $\overline{T}$ and σ_{T} are the mean and standard deviation of these intervals. Synchrony between two cells can be measured by calculating the *correlation coefficient*

$$C_{\mathrm{C}} = \frac{\mathrm{cov}(n_1, n_2)}{\sqrt{\mathrm{var}(n_1)\mathrm{var}(n_2)}} \tag{4}$$

of their spike trains n_1 and n_2, respectively (Perkel et al., 1967). The variance (var) and covariance (cov) are calculated by using time bins with 2 ms duration (Kumar et al., 2008).

Brüderle et al. (2010) have shown another approach to investigate networks inspired by Brunel (2000). Their focus have been the effects of network parameters and STP on the firing rate of the network. In our study, we show that such BRNs can show an asynchronous irregular network state, when emulated on hardware.

3.2.2 Hardware Emulation

In addition to standard calibration routines (Section 2.4), we have calibrated the chip explicitly for the BRN shown in Figure 5A. In the first of two steps, excitatory and inhibitory synapse line drivers were calibrated sequentially towards equal strength, respectively, but with inhibition four times stronger than excitation. To this end, all available neurons received spiking activity from a single synapse line driver, thereby averaging out neuron-to-neuron variations. The shape of synaptic conductances (specifically t_{fall} and $g_i^{\max}$) were adjusted to obtain a target mean firing rate of 10 Hz over all neurons. Similarly, each driver was calibrated for its inhibitory operation mode. All neurons were strongly stimulated by an additional driver with its excitatory mode already calibrated, and again the shape of conductances, this time for inhibition, was adjusted to obtain the target rate.

Untouched by this prior calibration towards a target mean rate, neuron excitability still varied between neurons and was calibrated consecutively for each neuron in a second calibration step. For this, all neurons of the BRN were used to stimulate a single neuron with a total firing rate that was uniformly distributed among all inputs and equal to the estimated firing rate of the final network implementation. Subsequently, all afferent synaptic weights to this neuron were scaled in order to adapt its firing rate to the target rate.

To avoid a self-reinforcement of network activity observed in emulations on the hardware, efferent connections of the excitatory neuron population were modeled as short-term depressing. Nevertheless, such BRNs still show an asynchronous irregular network state (Figure 5B).

Figure 5C show recordings of a BRN emulation on a calibrated chip with neurons firing irregularly and asynchronously. Note that $C_{\mathrm{V}} \geq 1$ does not necessarily guarantee an exponential interspike interval distribution and even less Poisson firing. However, neurons within the BRN clearly exhibit irregular firing (compare raster plots of Figure 5B and C).

A simulation of the same network topology and stimulus using software tools produced similar results. Synaptic weights were not known for the hardware emulation, but defined

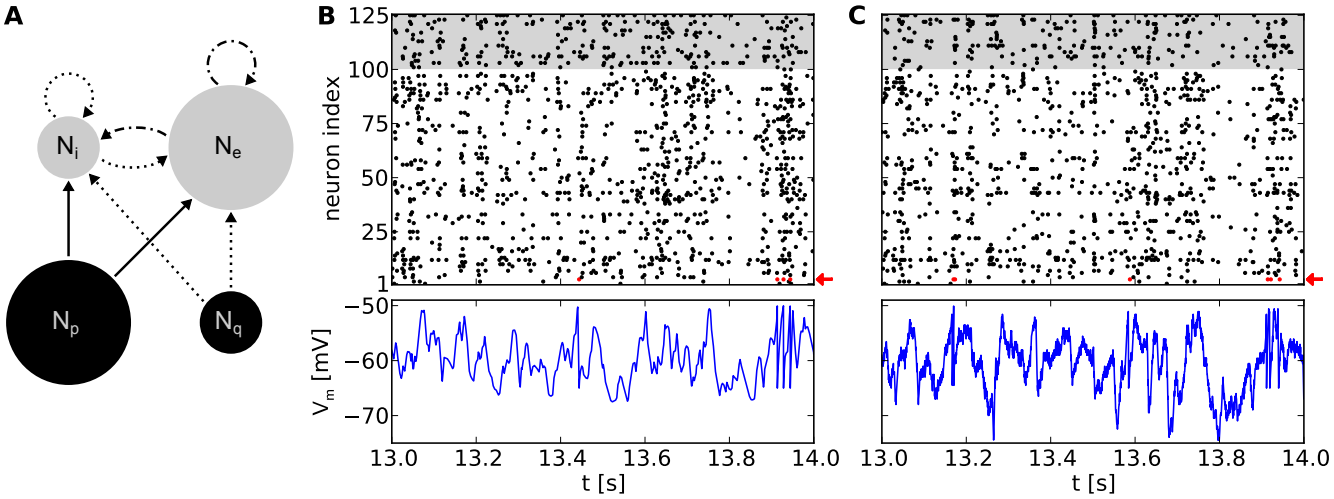


Figure 5: **(A)** Network topology of a balanced random network. Populations consisting of $N_\mathrm{e} = 100$ excitatory and $N_\mathrm{i} = 25$ inhibitory neurons (gray circles), respectively, are stimulated by populations of Poisson sources (black circles). We use $N_\mathrm{p} = 100$ independent sources for excitation and $N_\mathrm{q} = 25$ for inhibition. Arrows denote projections between these populations with connection probabilities $p = 0.1$, with solid lines for excitatory and dotted lines for inhibitory connections. Dot and dash lines are indicating excitatory projections with short-term depression. **(B)** Top: Raster plot of a software simulation. Populations of excitatory and inhibitory neurons are depicted with white and gray background, respectively. Note that for clarity only the time interval [13 s, 14 s] of a 20 s emulation is shown. For the full 20 s emulation, we have measured $C_\mathrm{V} = 0.96 \pm 0.09$ (mean over all neurons) and $C_\mathrm{C} = 0.010 \pm 0.017$ (mean over 1000 random chosen pairs of neurons), respectively. Bottom: Recorded membrane potential of an arbitrary excitatory neuron (neuron index 3, highlighted with a red arrow in the above raster plot). **(C)** Same network topology and stimulus as in (B), but emulated on the *Spikey* chip, resulting in $C_\mathrm{V} = 1.02 \pm 0.16$ and $C_\mathrm{C} = 0.014 \pm 0.019$. Note that the membrane recordings are calibrated such that the threshold and reset potential match those of the software counterpart.

by the target firing rates using the above calibration. A translation to biological parameters is possible, but would have required further measurements and was not of further interest in this context. Instead, for software simulations, the synaptic weight for excitatory connections were chosen to fit the mean firing rate of the hardware emulation (approx. 9 Hz). Then, the weight of inhibitory connections were chosen to preserve the ratio between inhibitory and excitatory weights.

Membrane dynamics of single neurons within the network are comparable between hardware emulations and software simulations (Figure 5B and C). Evidently, spike times differ between the two approaches due to various hardware noise sources (Section 2.4). However, in "large" populations of neurons ($Ne + Ni = 125$ neurons), we observe that these phenomena have qualitatively no effect on firing statistics, which are comparable to software simulations (compare raster plots of Figure 5B and C). The ability to reproduce these statistics is highly relevant in the context of cortical models which rely on asynchronous irregular firing activity for information processing (e.g., van Vreeswijk & Sompolinsky, 1996).

3.3 Soft Winner-Take-All Network

Soft winner-take-all (sWTA) computation is often viewed as an underlying principle in models of cortical processing (Grossberg, 1973; Maass, 2000; Itti & Koch, 2001; Douglas & Martin, 2004; Oster et al., 2009; Lundqvist et al., 2010). The sWTA architecture has many practical applications, for example contrast enhancement, or making a decision which of two concurrent inputs is larger. Many neuromorphic systems explicitly implement sWTA architectures (Lazzaro et al., 1988; Chicca et al., 2007; Neftci et al., 2011).

3.3.1 Network Topology

We implemented an sWTA network that is composed of a ring-shaped layer of recurrently connected excitatory and a common pool of inhibitory neurons (Figure 6A), following the implementation by Neftci et al. (2011). Excitatory neurons project to the common inhibitory pool and receive recurrent feedback from there. In addition, excitatory neurons have recurrent excitatory connections to their neighbors on the ring. The strength of these decays with increasing distance on the ring, following a Gaussian profile with a standard deviation of $\sigma_{\text{rec}} = 5$ neurons. External stimulation is also received through a Gaussian profile, with the mean μ_{ext} expressing the neuron index that receives input with maximum synaptic strength. Synaptic input weights to neighbors of that neuron decay according to a standard deviation of $\sigma_{\text{ext}} = 3$ neurons. We clipped the input weights to zero beyond $\sigma_{\text{ext}} \cdot 3$. Each neuron located within the latter Gaussian profile receives stimulation from five independent Poisson spike sources each firing at rate r. Depending on the contrast between the input firing rates r_1 and r_2 of two stimuli applied to opposing sides of the ring, one side of the ring "wins" by firing with a higher rate and thereby suppressing the other.

3.3.2 Hardware Emulation

We assessed the efficiency of this sWTA circuit by measuring the reduction in firing rate exerted in neurons when the opposite side of the ring is stimulated. We stimulated one side of the ring with a constant, and the opposite side with a varying firing rate. In case of hardware emulations, each stimulus was distributed and hence averaged over multiple line drivers in order to equalize stimulation strength among neurons. For both back-ends, inhibitory weights were chosen four times stronger than excitatory ones (using the synapse line driver calibration of Section 3.2).

The firing rate of the reference side decreased when the firing rate of stimulation to the opposite side was increased, both in software simulation and on the hardware (Figure 6B and C). In both cases, the average firing rates crossed at approximately $r_2 = 50\,\text{Hz}$, corresponding to the spike rate delivered to the reference side. The firing rates r_{tot} are less distinctive for hardware emulations compared to software simulations, but still sufficient to produce robust sWTA functionality. Note that the observed firing rates are higher on the hardware than in the software simulation. This difference is due to the fact that the reliability of the network performance improved for higher firing rates.

Figure 6D and E depict activity profiles of the excitatory neuron layer. The hardware neurons exhibited a broader and also slightly asymmetric excitation profile compared to the software simulation. The asymmetry is likely due to inhomogeneous excitability of neurons, which is caused by fixed-pattern noise (Section 2). The broader excitation profile indicates that inhibition is less efficient on the hardware than in the software simulation (a trend that can also be observed in the firing rates in Figure 6B and C). Counteracting this

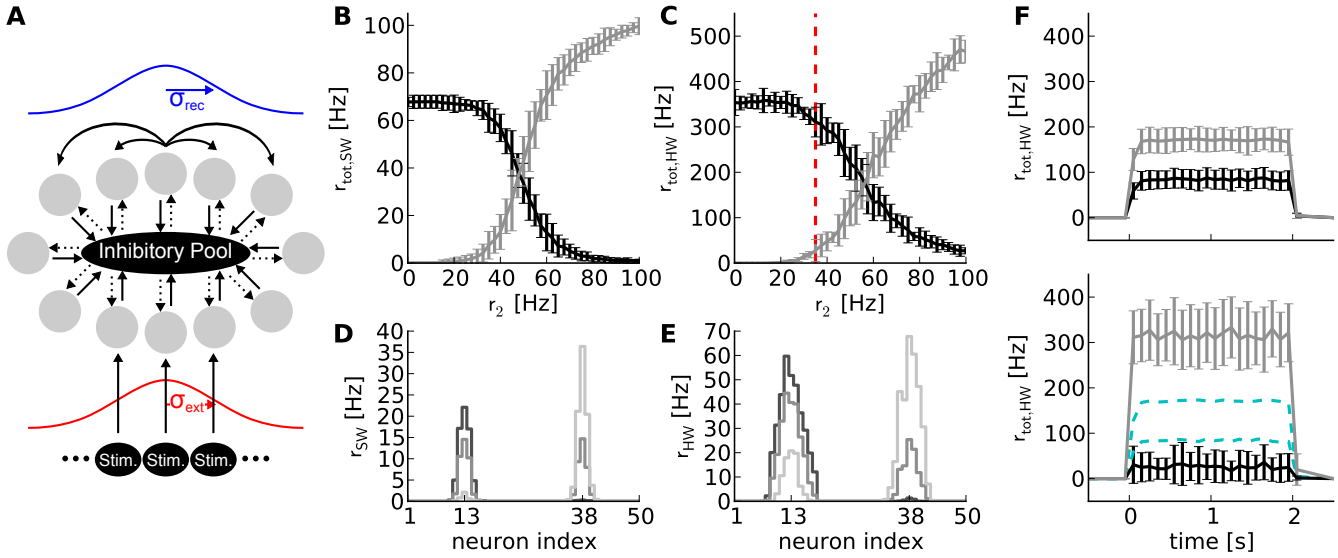


Figure 6: **(A)** Topology of a soft winner-take-all network with 50 excitatory (gray circles) and 16 inhibitory neurons. Solid and dotted arrows denote excitatory and inhibitory connections, respectively. The strength profile of recurrent connections between excitatory neurons and external stimulations is schematized in blue and red, respectively (for details see text). All projections between neuron populations have a connection probabilities of $p = 1$, except the projection between the excitatory and inhibitory neuron population ($p = 0.6$). **(B)** Results of software simulation (SW). Black curve: Total firing rate of the reference half where constant external stimulation is received ($r_1 = 50\,\text{Hz}$ at $\mu_{\text{ext}} =$ neuron index 13). Gray curve: Total firing rate of the neurons in the half of the ring where varying external stimulation with rate r_2 between zero and 100 Hz is received (at $\mu_{\text{ext}} =$ neuron index 38). Firing rates r_{tot} of all neurons in each half of the ring were averaged over 10 runs with 2 s duration and different random number seeds for drawing the stimulus spike trains. **(C)** Same network topology and stimulus as (B), but emulated on *Spikey* (HW). **(D)** Firing rate distribution over neuron indices for $r_2 = 25\,\text{Hz}$ (black), 50 Hz (dark gray) and 75 Hz (light gray). **(E)** Same as (D), but emulated on *Spikey*. **(F)** Top panel: Time course of firing rates for stimulus indicated in (C, red dashed line), but without recurrent connections. All excitatory neurons are solely driven by an external stimulus of $r_1 = 50\,\text{Hz}$ and $r_2 = 35\,\text{Hz}$, respectively. Firing rates were averaged over 100 runs. Bottom panel: Same as top panel, but with recurrent connections. For better comparison, data of the top panel is drawn in cyan dashed lines.

loss of inhibition may be possible through additional calibration, if the sharpness of the excitation profile is critical for the task in which such an sWTA circuit is to be employed.

The network emulated on *Spikey* is said to perform sWTA, because the side of the ring with stronger stimulation shows an amplified firing rate, while the firing rate of the other side is suppressed (see Figure 6F). This qualifies our hardware system for applications relying on similar sWTA network topologies.

3.4 Cortical Layer 2/3 Attractor Model

Throughout the past decades, attractor networks that model working memory in the cerebral cortex have gained increasing support from both experimental data and computer simulations. The *cortical layer 2/3 attractor memory model* described in Lundqvist et al. (2006, 2010) has been remarkably successful at reproducing both low-level (firing patterns, membrane potential dynamics) and high level (pattern completion, attentional blink) features of cortical information processing. One particularly valuable aspect is the very low amount of fine-tuning this model requires in order to reproduce the rich set of desired internal dynamics. It has also been shown in Brüderle et al. (2011) that there are multiple ways of scaling this model down in size without affecting its main functionality features. These aspects make it an ideal candidate for implementation on our analog neuromorphic device. In this context, it becomes particularly interesting to analyze how the strong feedback loops which predominantly determine the characteristic network activity are affected by the imposed limitations of the neuromorphic substrate and fixed-pattern noise. Here, we extend the work done in Brüderle et al. (2011) by investigating specific attractor properties such as firing rates, voltage UP-states and the pattern completion capability of the network.

3.4.1 Network Topology

From a structural perspective, the most prominent feature of the Layer 2/3 Attractor Memory Network is its modularity. Faithful to its biological archetype, it implements a set of cortical hypercolumns, which are in turn subdivided into multiple minicolumns (Figure 7A). Each minicolumn consists of three cell populations: excitatory pyramidal cells, inhibitory basket cells and inhibitory RSNP (regular spiking non-pyramidal) cells.

Attractor dynamics arise from the synaptic connectivity on two levels. Within a hypercolumn, the basket cell population enables a soft-WTA-like competition among the pyramidal populations within the minicolumns. On a global scale, the long-range inhibition mediated by the RSNP cells governs the competition among so-called *patterns*, as explained in the following.

In the original model described in Lundqvist et al. (2010), each hypercolumn contains 9 minicolumns, each of which consists of 30 pyramidal, 2 RSNP and 1 basket cells. Within a minicolumn, the pyramidal cells are interconnected and also project onto the 8 closest basket cells within the same hypercolumn. In turn, pyramidal cells in a minicolumn receive projections from all basket cells within the same hypercolumn. All pyramidal cells receive two types of additional excitatory input: an evenly distributed amount of diffuse Poisson noise and specific activation from the cortical layer 4. Therefore, the minicolumns (i.e., the pyramidal populations within) compete among each other in WTA-like fashion, with the winner being determined by the overall strength of the received input.

A pattern (or attractor) is defined as containing exactly one minicolumn from each hypercolumn. Considering only orthogonal patterns (each minicolumn may only belong to a single pattern) and given that all hypercolumns contain an equal amount of mini-

columns, the number of patterns in the network is equal to the number of minicolumns per hypercolumn. Pyramidal cells within each minicolumn project onto the pyramidal cells of all the other minicolumns in the same pattern. These connections ensure a spread of local activity throughout the entire pattern. Additionally, the pyramidal cells also project onto the RSNP cells of all minicolumns belonging to different attractors, which in turn inhibit the pyramidal cells within their minicolumn. This long-range competition enables the winning pattern to completely shut down the activity of all other patterns.

Two additional mechanisms weaken active patterns, thereby facilitating switches between patterns. The pyramidal cells contain an adaptation mechanism which decreases their excitability with every emitted spike. Additionally, the synapses between pyramidal cells are modeled as short-term depressing.

3.4.2 Hardware Emulation

When scaling down the original model (2673 neurons) to the maximum size available on the *Spikey* chip (192 neurons, see Figure 7B for software simulation results), we made use of the essential observation that the number of pyramidal cells can simply be reduced without compensating for it by increasing the corresponding projection probabilities. Also, for less than 8 minicolumns per hypercolumn, all basket cells within a hypercolumn have identical afferent and efferent connectivity patterns, therefore allowing to treat them as a single population. Their total number was decreased, while increasing their efferent projection probabilities accordingly. In general (i.e., except for pyramidal cells), when number and/or size of populations were changed, projection probabilities were scaled in such a way that the total fan-in for each neuron was kept at a constant average. When the maximum fan-in was reached (one afferent synapse for every neuron in the receptive field), the corresponding synaptic weights were scaled up by the remaining factor.

Because neuron and synapse models on the *Spikey* chip are different to the ones used in the original model, we have performed a heuristic fit in order to approximately reproduce the target firing patterns. Neuron and synapse parameters were first fitted in such a way as to generate clearly discernible attractors with relatively high average firing rates (see Figure 7D). Additional tuning was needed to compensate for missing neuronal adaptation, limitations in hardware configurability, parameter ranges and fixed-pattern noise affecting hardware parameters.

During hardware emulations, apart from the appearance of spontaneous attractors given only diffuse Poisson stimulation of the network (Figure 7C), we were able to observe two further interesting phenomena which are characteristic for the original attractor model.

When an attractor becomes active, its pyramidal cells enter a so-called UP state which is characterized by an elevated average membrane potential. Figure 7E clearly shows the emergence of such UP-states on hardware. The onset of an attractor is characterized by a steep rise in pyramidal cell average membrane voltage, which then decays towards the end of the attractor due to synaptic short-term depression and/or competition from other attractors temporarily receiving stronger stimulation. On both flanks of an UP state, the average membrane voltage shows a slight undershoot, due to the inhibition by other active attractors.

A second important characteristic of cortical attractor models is their capability of performing *pattern completion* (Lundqvist et al., 2006). This means that a full pattern can be activated by stimulating only a subset of its constituent pyramidal cells (in the original model, by cells from cortical Layer 4, modeled by us as additional Poisson sources). The appearance of this phenomenon is similar to a phase transition from a resting state to a collective pyramidal UP-state occurring when a critical amount of pyramidal cells are

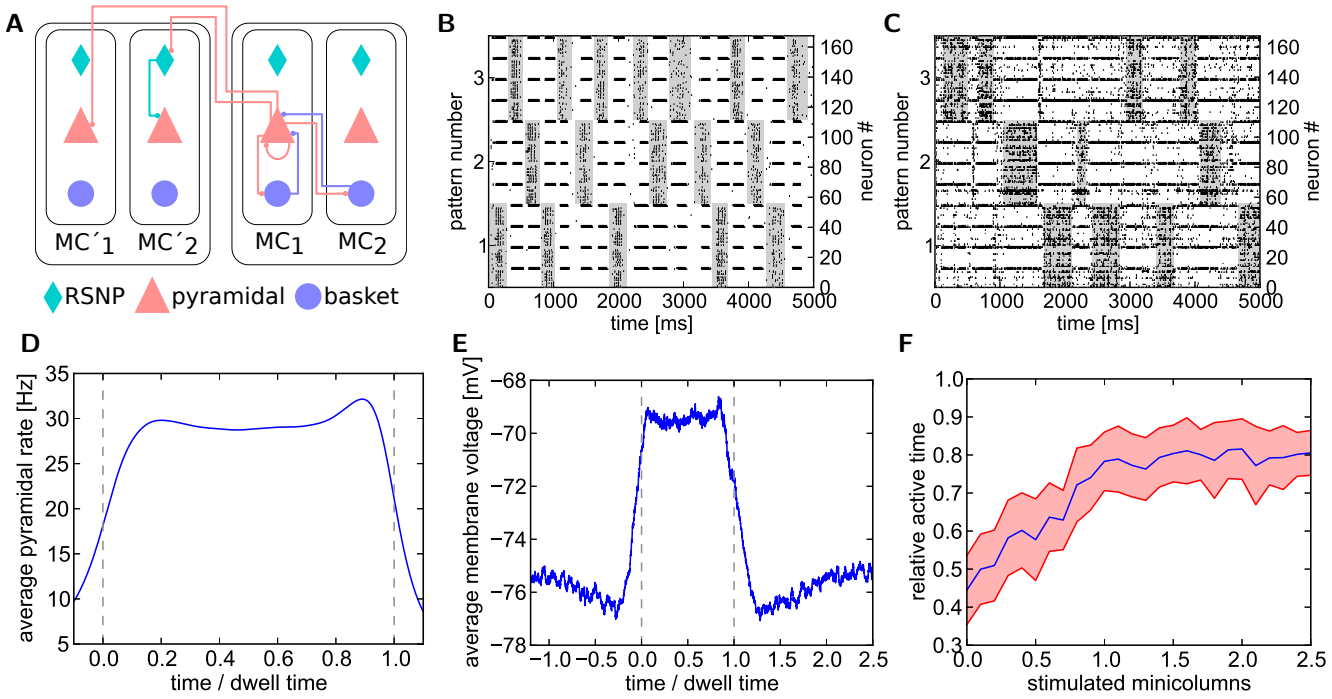


Figure 7: **(A)** Schematic of the cortical layer 2/3 attractor memory network. Two hypercolumns, each containing two minicolumns, are shown. For better readability, only connections that are active within an active pattern are depicted. See text for details. **(B)** Software simulation of spiking activity in the cortical attractor network model scaled down to 192 neurons (only pyramidal and RSNP cells shown, basket cells spike almost continously). Minicolumns belonging to the same pattern are grouped together. The broad stripes of activity are generated by pyramidal cells in active attractors. The interlaced narrow stripes of activity represent pairs of RSNP cells, which spike when their home minicolumn is inhibited by other active patterns. **(C)** Same as **B**, but on hardware. The raster plot is noisier and the duration of attractors (dwell time) are less stable than in software due to fixed-pattern noise on neuron and synapse circuits. For better readability, active states are underlied in grey in **B** and **C**. **(D)** Average firing rate of pyramidal cells on the *Spikey* chip inside active patterns. To allow averaging over multiple active periods of varying lengths, all attractor dwell times have been normalized to 1. **(E)** Average membrane potential of pyramidal cells on the *Spikey* chip inside and outside active patterns. **(F)** Pattern completion on the *Spikey* chip. Average values (from multiple runs) depicted in blue, with the standard deviation shown in red. From a relatively equilibrated state where all patterns take turns in being active, additional stimulation (see text) of only a subset of neurons from a given attractor activates the full pattern and enables it to dominate over the other two. The pattern does not remain active indefinitely due to short-term depression in excitatory synapses, thereby still allowing short occasional activations of the other two patterns.

stimulated. To demonstrate pattern completion, we have used the same setup as in the previous experiments, except for one pattern receiving additional stimulation. From an initial equilibrium between the three attractors (approximately equal active time), we have observed the expected sharp transition to a state where the stimulated attractor dominates the other two, occurring when one of its four minicolumns received L4 stimulus (Figure 7F).

The implementation of the attractor memory model is a particularly comprehensive showcase of the configurability and functionality of our neuromorphic platform due to the complexity of both model specifications and emergent dynamics. Starting from these results, the next-generation hardware (Schemmel et al., 2010) will be able to much more accurately model biological behavior, thanks to a more flexible, adapting neuron model and a significantly increased network size.

3.5 Insect Antennal Lobe Model

The high acceleration factor of the *Spikey* chip makes it an attractive platform for neuromorphic data processing. Preprocessing of multivariate data is a common problem in signal and data analysis. In conventional computing, reduction of correlation between input channels is often the first step in the analysis of multidimensional data, achieved, e.g., by *principal component analysis* (PCA). The architecture of the olfactory system maps particularly well onto this problem (Schmuker & Schneider, 2007). We have implemented a network that is inspired by processing principles that have been described in the insect antennal lobe (AL), the first relay station from olfactory sensory neurons to higher brain areas. The function of the AL has been described to decorrelate the inputs from sensory neurons, potentially enabling more efficient memory formation and retrieval (Stopfer et al., 1997; Linster & Smith, 1997; Perez-Orive et al., 2004; Wilson & Laurent, 2005). The mammalian analog of the AL (the olfactory bulb) has been the target of a recent neuromorphic modeling study (Imam et al., 2012b).

The availability of a network building block that achieves channel decorrelation is an important step toward high-performance neurocomputing. The aim of this experiment is to demonstrate that the previously studied rate-based AL model (Schmuker & Schneider, 2007) that reduces rate correlation between input channels is applicable to a spiking neuromorphic hardware system.

3.5.1 Network Topology

In the insect olfactory system, odors are first encoded into neuronal signals by receptor neurons (RNs) which are located on the antenna. RNs send their axons to the AL (Figure 8A). The AL is composed of glomeruli, spherical compartments where RNs project onto local inhibitory neurons (LNs) and projection neurons (PNs). LNs project onto other glomeruli, effecting lateral inhibition. PNs relay the information to higher brain areas where multimodal integration and memory formation takes place.

The architecture of our model reflects the neuronal connectivity in the insect AL (Figure 8A). RNs are modeled as spike train generators, which project onto the PNs in the corresponding glomerulus. The PNs project onto the LNs, which send inhibitory projections to the PNs in other glomeruli.

In biology, the AL network reduces the rate correlation between glomeruli, in order to improve stimulus separability and thus odor identification. Another effect of decorrelation is that the rate patterns encoding the stimuli become sparser, and use the available coding space more efficiently as redundancy is reduced. Our goal was to demonstrate the reduc-

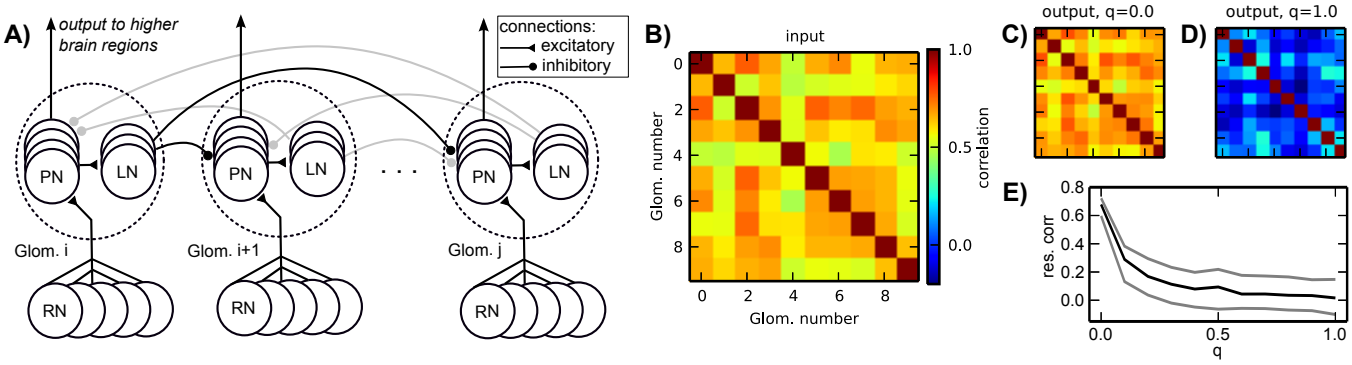


Figure 8: **(A)** Schematic of the insect antennal lobe network. Neuron populations are grouped in glomeruli (outlined by dotted lines), which exert lateral inhibition onto each other. RNs: receptor neurons (input), PNs: projection neurons (output), LNs: inhibitory local neurons. Some connections are grayed out to emphasize the connection principle. **(B)** Correlation matrix of the input data. **(C)** Correlation matrix of the output spike rates (PNs) without lateral inhibition, $q = 0.0$. **(D)** Correlation of the output with homogeneous lateral inhibition, $q = 1.0$. **(E)** Average pairwise correlation between glomeruli (median $\pm$ 20^{th} (black) and 80^{th} (gray) percentile) in dependence of the overall strength of lateral inhibition q.

tion of rate correlations across glomeruli (*channel correlation*) by the AL-inspired spiking network. To this end, we generated patterns of firing rates with channel correlation. We created a surrogate data set exhibiting channel correlation using a copula, a technique that allows to generate correlated series of samples from an arbitrary random distribution and a covariance matrix (Nelsen, 1998). The covariance matrix was uniformly set to a target correlation of 0.6. Using this copula, we sampled 100 ten-dimensional data vectors from an exponential distribution. In the biological context, this is equivalent to having a repertoire of 100 odors, each encoded by ten receptors, and the firing rate of each input channel following a decaying exponential distribution. Values larger than e were clipped and the distribution was mapped to the interval $[0, 1]$ by applying $v = v/e$ for each value v. These values were then converted into firing rates between 20 and 55 spikes/s. The ten-dimensional data vector was presented to the network by mapping the ten firing rates onto the ten glomeruli, setting all single RNs in each glomerulus to fire at the respective target rates. Rates were converted to spike trains individually for each RN using the Gamma process with $\gamma = 5$. Each data vector was presented to the network for the duration of one second by making the RNs of each glomerulus fire with the specified rate. The inhibitory weights between glomeruli were uniform, i.e., all inhibitory connections shared the same weight. During one second of stimulus presentation, output rates were measured from PNs. One output rate per glomerulus was obtained by averaging the firing rate of all PNs in a glomerulus.

We have used 6 RN input streams per glomerulus, projecting in an all-to-all fashion onto 7 PNs, which in turn projected on 3 LNs per glomerulus.

3.5.2 Hardware Emulation

The purpose of the presented network was to reduce rate correlation between input channels. As in other models, fixed-pattern noise across neurons had a detrimental effect on the function of the network. We exploited the specific structure of our network to im-

plement more efficient calibration than can be provided by standard calibration methods (Section 2.4). Our calibration algorithm targeted PNs and LNs in the first layer of the network. During calibration, we turned off all projections between glomeruli. Its aim was to achieve a homogeneous response across PNs and LNs respectively, i.e., within ± 10% of a target rate. The target rate was chosen from the median response rate of uncalibrated neurons. For neurons whose response rate was too high it was sufficient to reduce the synaptic weight of the excitatory input from RNs. For those neurons with a too low rate the input strength had to be increased. The excitatory synaptic weight of the input from RNs was initially already at its maximum value and could not be increased. As a workaround we used PNs from the same glomerulus to add additional excitatory input to those "weak" neurons. We ensured that no recurrent excitatory loops were introduced by this procedure. If all neurons in a glomerulus were too weak, we recruit another external input stream to achieve the desired target rate. Once the PNs were successfully calibrated (less than 10% deviation from the target rate), we used the same approach to calibrate the LNs in each glomerulus.

To assess the performance of the network we have compared the channel correlation in the input and in the output. The channel correlation matrix $\mathbf{C}$ was computed according to

$$C_{i,j} = d^{\text{Pearson}}(\boldsymbol{\nu}_{\text{glom}.i}, \boldsymbol{\nu}_{\text{glom}.j}) \,, \tag{5}$$

with $d^{\text{Pearson}}(\cdot, \cdot)$ the Pearson correlation coefficient between two vectors. For the input correlation matrix $\mathbf{C}^{\text{input}}$, the vector $\boldsymbol{\nu}_{\text{glom}.i}$ contained the average firing rates of the six RNs projecting to the ith glomerulus, with each element of this vector for one stimulus presentation. For the output correlation matrix $\mathbf{C}^{\text{output}}$ we used the rates from PNs instead of RNs. Thus, we obtained 10×10 matrices containing the rate correlations for each pair of input or output channels.

Figure 8B depicts the correlation matrix $\mathbf{C}^{\text{input}}$ for the input firing rates. When no lateral inhibition is present, $\mathbf{C}^{\text{input}}$ matches $\mathbf{C}^{\text{output}}$ (Figure 8C). We have systematically varied the strength of lateral inhibition by scaling all inhibitory weights by a factor q, with $q = 0$ for zero lateral inhibition and $q = 1$ for inhibition set to its maximal strength. With increasing lateral inhibition, off-diagonal values in $\mathbf{C}^{\text{output}}$ approach zero and output channel correlation is virtually gone (Figure 8D). The amount of residual correlation to be present in the output can be controlled by adjusting the strength of lateral inhibition (Figure 8E).

Taken together, we demonstrated the implementation of an olfaction-inspired network to remove correlation between input channels on the *Spikey* chip. This network can serve as a preprocessing module for data analysis applications to be implemented on the *Spikey* chip. An interesting candidate for such an application is a spiking network for supervised classification, which may benefit strongly from reduced channel correlations for faster learning and better discrimination (Häusler et al., 2011).

3.6 Liquid State Machine

Liquid state machines (LSMs) as proposed by Maass et al. (2002) and Jaeger (2001) provide a generic framework for computation on continuous input streams. The *liquid*, a recurrent network, projects an input into a high-dimensional space which is subsequently read out. It has been proven that LSMs have universal computational power for computations with fading memory on functions of time (Maass et al., 2002). In the following, we show that classification performance of an LSM emulated on our hardware is comparable to the corresponding computer simulation. Synaptic weights of the readout are iteratively learned

on-chip, which inherently compensates for fixed-pattern noise. A trained system can then be used as an autonomous and very fast spiking classifier.

3.6.1 Network Topology

The LSM consists of two major components: the recurrent liquid network itself and a spike-based classifier (Figure 9A). A general purpose liquid needs to meet the separation property (Maass et al., 2002), which requires that different inputs are mapped to different outputs, for a wide range of possible inputs. Therefore, we use a network topology similar to the one proposed by Bill et al. (2010). It consists of an excitatory and inhibitory population with a ratio of 80:20 excitatory to inhibitory neurons. Both populations have recurrent as well as feedforward connections. Each neuron in the liquid receives 4 inputs from the 32 excitatory and 32 inhibitory sources, respectively. All other connection probabilities are illustrated in Figure 9.

The readout is realized by means of a tempotron (Gütig & Sompolinsky, 2006), which is compatible with our hardware due to its spike-based nature. Furthermore, its modest single neuron implementation leaves most hardware resources to the liquid. The afferent synaptic weights are trained with the method described in Gütig & Sompolinsky (2006), which effectively implements gradient descent dynamics. Upon training, the tempotron distinguishes between two input classes by emitting either one or no spike within a certain time window. The former is artificially enforced by blocking all further incoming spikes after the first spike occurrence.

The PSP kernel of a LIF neuron with current-based synapses is given by

$$K(t-t_i) = A\left(e^{-\frac{t-t_i}{\tau_\mathrm{m}}} - e^{-\frac{t-t_i}{\tau_\mathrm{s}}}\right) \cdot \Theta(t-t_i)\ , \tag{6}$$

with the membrane time constant τ_m and the synaptic time constant τ_s, respectively. Here, A denotes a constant PSP scaling factor, t_i the time of the ith incoming spike and $\Theta(t)$ the Heaviside step function.

During learning, weights are updated as follows

$$\Delta w_j^n = \begin{cases} 0 & \text{correct} \\ \alpha(n) \sum_{t_{i,j}<t_\mathrm{max}} K(t_\mathrm{max} - t_{i,j}) & \text{erroneous,} \end{cases} \tag{7}$$

where Δw_j^n is the weight update corresponding to the jth afferent neuron after the nth learning iteration with learning rate $\alpha(n)$. The spike time of the tempotron, or otherwise the time of highest membrane potential, is denoted with t_max. In other words, for trials where an erroneous spike was elicited, the excitatory afferents with a causal contribution to this spike are weakened and inhibitory ones are strengthened according to Equation 7. In case the tempotron did not spike even though it should have, the weights are modulated the other way round, i.e. excitatory weights are strengthened and inhibitory ones are weakened. This learning rule has been implemented on hardware with small modifications, due to the conductance-based nature of the hardware synapses (see below).

The tempotron is a binary classifier, hence any task needs to be mapped to a set of binary decisions. Here, we have chosen a simple binary task adapted from Maass et al. (2002), to evaluate the performance of the LSM. The challenge was to distinguish spike train segments in a continuous data stream composed of two templates with identical rates (denoted X and Y in Figure 9A). In order to generate the input, we cut the template spike trains into segments of 50 ms duration. We then composed the spike sequence to be presented to the network by randomly picking a spike segment from either X or Y in each

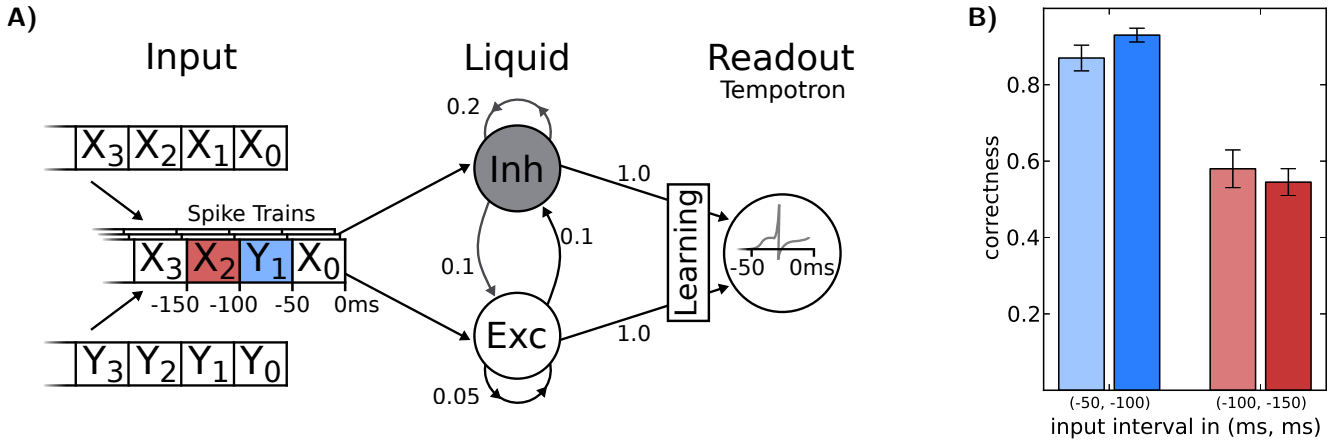


Figure 9: **(A)** Schematic of the LSM and the given task. Spike sources are composed of 50 ms segments drawn from two template spike trains (X and Y). These patterns are streamed into the liquid (with descending index), which is a network consisting of 191 neurons, leaving one neuron for the tempotron. Connection probabilities are depicted next to each connection (arrows). In two experiments, the tempotron is trained to either classify the origin (X or Y) of the spike train segment with index 1 or 2. **(B)** The classification performance of the LSM measured over 200 samples after 1000 training iterations for both hardware (lighter) and software (darker) implementation.

time window (see Figure 9 for a schematic). Additionally, we added spike timing jitter from a normal distribution with a standard deviation of $\sigma = 1$ ms to each spike. For each experiment run, both for training and evaluation, the composed spike sequence was then streamed into the liquid. Tempotrons were given the liquid activity as input and trained to identify whether the segment within the previous time window originated from sequence X or Y. In a second attempt, we trained the tempotron to identify the origin of the pattern presented in the window at -100 to -150 ms (that is, the second to the last window). Not only did this task allow to determine the classification capabilities of the LSM, but it also put the liquid's fading memory to the test, as classification of a segment further back in time becomes increasingly difficult.

3.6.2 Hardware Emulation

The liquid itself does not impose any strong requirements on the hardware since virtually any network is suitable as long as the separation property is satisfied. We adapted a network from Bill et al. (2010) which, in a similar form, had already been implemented on our hardware. However, STP was disabled, because at the time of the experiment it was not possible to exclusively enable STP for the liquid without severely affecting the performance of the tempotron.

The hardware implementation of the tempotron required more attention, since only conductance-based synapses are available. The dependence of spike efficacies on the actual membrane potential was neglected, because the rest potential was chosen to be close to the firing threshold, with the reversal potentials far away. However, the asymmetric distance of excitatory and inhibitory reversal potentials from the sub-threshold regime needed compensation. This was achieved by scaling all excitatory weights by $(\overline{V_\mathrm{m}} - E_\mathrm{inh})/(\overline{V_\mathrm{m}} - E_\mathrm{exc})$, where $\overline{V_\mathrm{m}}$ corresponds to the mean neuron membrane voltage and $E_\mathrm{exc}/E_\mathrm{inh}$ is the excitatory/inhibitory reversal potentials. Discontinuities in spike efficacies for synapses changing

from excitatory to inhibitory or vice versa were avoided by prohibiting such transitions. Finally, membrane potential shunting after the first spike occurrence is neither possible on our hardware nor very biological and had therefore been neglected, as already proposed by Gütig & Sompolinsky (2006).

Even though the tempotron was robust against fixed-pattern noise due to on-chip learning, the liquid required modifications. Therefore, firing thresholds were tuned independently in software and hardware to optimize the memory capacity and avoid violations of the separation property. Since hardware neurons share firing thresholds, the tempotron was affected accordingly (see Table 1). Additionally, the learning curve $\alpha(n)$ was chosen individually for software and hardware due to the limited resolution of synaptic weights on the latter.

The results for software and hardware implementations are illustrated in Figure 9B. Both LSMs performed at around 90% classification correctness for the spiketrain segment that lied 50 ms to 100 ms in the past with respect to the end of the stimulus. For inputs lying even further away in time, performances dropped to chance level (50% for a binary task), independent of the simulation back-end.

Regarding the classification capabilities of the LSM, our current implementation allows a large variety of tasks to be performed. Currently, e.g., we are working on hand-written digit recognition with the very same setup on the *Spikey* chip. Even without a liquid, our implementation of the tempotron (or populations thereof) makes an excellent neuromorphic classifier, given its bandwidth-friendly sparse response and robustness against fixed-pattern noise.

4 Discussion

We have successfully implemented a variety of neural microcircuits on a single universal neuromorphic substrate, which is described in detail by Schemmel et al. (2006). All networks show activity patterns qualitatively and to some extent also quantitatively similar to those obtained by software simulations. The corresponding reference models found in literature have not been modified significantly and network topologies have been identical for hardware emulation and software simulation, if not stated otherwise. In particular, the emulations benefit from the advantages of our neuromorphic implementation, namely inherent parallelism and accelerated operation compared to software simulations on conventional von-Neumann machines. Previous accounts of networks implemented on the *Spikey* system include computing with high-conductance states (Kaplan et al., 2009), self-stabilizing recurrent networks (Bill et al., 2010), and simple emulations of cortical layer 2/3 attractor networks (Brüderle et al., 2011).

In this contribution, we have presented a number of new networks and extensions of previous implementations. Our synfire chain implementation achieves reliable signal propagation over years of biological time from one single stimulation, while synchronizing and filtering these signals (Section 3.1). Our extension of the network from Bill et al. (2010) to exhibit asynchronous irregular firing behavior is an important achievement in the context of reproducing stochastic activity patterns found in cortex (Section 3.2). We have realized soft winner-take-all networks on our hardware system (Section 3.3), which are essential building blocks for many cortical models involving some kind of attractor states (e.g., the decision-making model by Soltani & Wang, 2010). The emulated cortical attractor model provides an implementation of working memory for computation with cortical columns (Section 3.4). Additionally, we have used the *Spikey* system for preprocessing of multivariate data inspired by biological archetypes (Section 3.5) and machine learning

(Section 3.6). Most of these networks allocate the full number of neurons receiving input from one synapse array on the *Spikey* chip, but with different sets of neuron and synapse parameters and especially vastly different connectivity patterns, thereby emphasizing the remarkable configurability of our neuromorphic substrate.

However, the translation of such models requires modifications to allow execution on our hardware. The most prominent cause for such modifications is fixed-pattern noise across analog hardware neurons and synapses. In most cases, especially when population rate coding is involved, it is sufficient to compensate for this variability by averaging spiking activity over many neurons. For the data decorrelation and machine learning models, we have additionally trained the synaptic weights on the chip to achieve finer equilibration of the variability at critical network nodes. Especially when massive downscaling is required in order for models to fit onto the substrate, fixed pattern noise presents an additional challenge because the same amount of information needs to be encoded by fewer units. For this reason, the implementation of the cortical attractor memory network required additional heuristic activity fitting procedures.

The usability of the *Spikey* system, especially for neuroscientists with no neuromorphic engineering background, is provided by an integrated development environment. We envision that the configurability made accessible by such a software environment will encourage a broader neuroscience community to use our hardware system. Examples of use would be the acceleration of simulations as well as the investigation of the robustness of network models against parameter variability, both between computational units and between trials, as e.g. published by Brüderle et al. (2010) and Schmuker et al. (2011). The hardware system can be efficiently used without knowledge about the hardware implementation on transistor level. Nevertheless, users have to consider basic hardware constraints, as e.g., shared parameters. Networks can be developed using the PyNN metalanguage and optionally be prototyped on software simulators before running on the *Spikey* system (Davison et al., 2009; Brüderle et al., 2009). This rather easy configuration and operation of the *Spikey* chip allows the implementation of many other neural network models.

There exist also boundaries to the universal applicability of our hardware system. One limitation inherent to this type of neuromorphic device is the choice of implemented models for neuron and synapse dynamics. Models requiring, e.g., neuronal adaptation or exotic synaptic plasticity rules are difficult, if not impossible to be emulated on this substrate. Also, the total number of neurons and synapses set a hard upper bound on the size of networks that can be emulated. However, the next generation of our highly accelerated hardware system will increase the number of available neurons and synapses by a factor of 10^3, and provide extended configurability for each of these units (Schemmel et al., 2010).

The main purpose of our hardware system is to provide a flexible platform for highly accelerated emulation of spiking neuronal networks. Other research groups pursue different design goals for their hardware systems. Some focus on dedicated hardware providing specific network topologies (e.g., Merolla & Boahen, 2006; Chicca et al., 2007), or comprising few neurons with more complex dynamics (e.g., Chen et al., 2010; Grassia et al., 2011; Brink et al., 2012). Others develop hardware systems of comparable configurability, but operate in biological real-time, mostly using off-chip communication (Vogelstein et al., 2007; Choudhary et al., 2012). Purely digital systems (Merolla et al., 2011; Furber et al., 2012; Imam et al., 2012a) and field-programmable analog arrays (FPAA; Basu et al., 2010) provide even more flexibility in configuration than our system, but have much smaller acceleration factors.

With the ultimate goal of brain size emulations, there exists a clear requirement for increasing the size and complexity of neuromorphic substrates. An accompanying upscal-

ing of the fitting and calibration procedures presented here appears impractical for such orders of magnitude and can only be done for a small subset of components. Rather, it will be essential to step beyond simulation equivalence as a quality criterion for neuromorphic computing, and to develop a theoretical framework for circuits that are robust against, or even exploit the inherent imperfections of the substrate for achieving the required computational functions.

Acknowledgments

We would like to thank Dan Husmann, Stefan Philipp, Bernhard Kaplan, and Moritz Schilling for their essential contributions to the neuromorphic platform, Johannes Bill, Jens Kremkow, Anders Lansner, Mikael Lundqvist and Emre Neftci for assisting with the hardware implementation of the network models, Oliver Breitwieser for data analysis and Venelin Petkov for characterisation measurements of the *Spikey* chip.

The research leading to these results has received funding by the European Union 6th and 7th Framework Programme under grant agreement no. 15879 (FACETS), no. 269921 (BrainScaleS) and no. 243914 (Brain-i-Nets). Michael Schmuker has received support from the german ministry for research and education (BMBF) to Bernstein Center for Computational Neuroscience Berlin (grant no. 01GQ1001D) and from Deutsche Forschungsgemeinschaft within SPP 1392 (grant no. SCHM 2474/1-1).

The main contributors for each section are denoted with author initials: neuromorphic system (AG & EM); synfire chain (PM); balanced random network (TP); soft winner-take-all network (TP); cortical attractor model (MP); antennal lobe model (MS); liquid state machine (SJ). All authors contributed to writing this paper.

References

Abeles, M., Hayon, G., & Lehmann, D. (2004). Modeling compositionality by dynamic binding of synfire chains. *J. Comput. Neurosci. 17*(2), 179–201.

Aertsen, A., Diesmann, M., & Gewaltig, M.-O. (1996). Propagation of synchronous spiking activity in feedforward neural networks. *J. Physiol. (Paris) 90*(3/4), 243–247.

Akay, M. (2007). *Handbook of neural engineering - Part II: Neuro-nanotechnology: Artificial implants and neural prosthesis.* New York, USA: Wiley-IEEE Press.

Aviel, Y., Mehring, C., Abeles, M., & Horn, D. (2003). On embedding synfire chains in a balanced network. *Neural Comput. 15*(6), 1321–1340.

Baker, S. N., Spinks, R., Jackson, A., & Lemon, R. N. (2001). Synchronization in monkey motor cortex during a precision grip task. I. Task-dependent modulation in single-unit synchrony. *J. Neurophysiol. 85*(2), 869–85.

Basu, A., Ramakrishnan, S., Petre, C., Koziol, S., Brink, S., & Hasler, P. (2010). Neural dynamics in reconfigurable silicon. *IEEE Trans. Biomed. Circuits Syst. 4*(5), 311–319.

Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci. 18*, 10464–10472.

Bill, J., Schuch, K., Brüderle, D., Schemmel, J., Maass, W., & Meier, K. (2010). Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity. *Front. Comput. Neurosci. 4*(129).

Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., & Degnan, B. (2012). A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Trans. Biomed. Circuits Syst. PP*(99), 1.

Brüderle, D., Bill, J., Kaplan, B., Kremkow, J., Meier, K., Müller, E., & Schemmel, J. (2010). Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, France. IEEE Press.

Brüderle, D., Müller, E., Davison, A., Muller, E., Schemmel, J., & Meier, K. (2009). Establishing a novel modeling tool: A Python-based interface for a neuromorphic hardware system. *Front. Neuroinformatics 3*(17).

Brüderle, D., Petrovici, M., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., Grübl, A., Wendt, K., Müller, E., Schwartz, M.-O., Husmann de Oliveira, D., Jeltsch, S., Fieres, J., Schilling, M., Müller, P., Breitwieser, O., Petkov, V., Muller, L., Davison, A. P., Krishnamurthy, P., Kremkow, J., Lundqvist, M., Muller, E., Partzsch, J., Scholze, S., Zühl, L., Destexhe, A., Diesmann, M., Potjans, T. C., Lansner, A., Schüffny, R., Schemmel, J., & Meier, K. (2011). A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol. Cybern. 104*, 263–296.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci. 8*(3), 183–208.

Carnevale, T., & Hines, M. (2006). *The NEURON Book*. Cambridge: Cambridge University Press.

Chen, H., Saïghi, S., Buhry, L., & Renaud, S. (2010). Real-time simulation of biologically realistic stochastic neurons in VLSI. *IEEE Trans. Neural Netw. 21*(9), 1511–1517.

Chicca, E., Whatley, A. M., Lichtsteiner, P., Dante, V., Delbruck, T., Del Giudice, P., Douglas, R. J., & Indiveri, G. (2007). A multichip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity. *IEEE Trans. Circuits Syst. I, Reg. Papers 54*(5), 981–993.

Choudhary, S., Sloan, S., Fok, S., Neckar, A., Trautmann, E., Gao, P., Stewart, T., Eliasmith, C., & Boahen, K. (2012). Silicon neurons that compute. In A. Villa, W. Duch, P. Érdi, F. Masulli, & G. Palm (Eds.), *Artificial Neural Networks and Machine Learning – ICANN 2012*, Volume 7552 of *Lecture Notes in Computer Science*, pp. 121–128. Springer Berlin / Heidelberg.

Davison, A., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., & Yger, P. (2009). PyNN: a common interface for neuronal network simulators. *Front. Neuroinformatics 2*(11).

Davison, A., Muller, E., Brüderle, D., & Kremkow, J. (2010). A common language for neuronal networks in software and hardware. *The Neuromorphic Engineer*.

Dayan, P., & Abbott, L. F. (2001). *Theoretical Neuroscience*. Cambridge: MIT Press.

Diesmann, M., Gewaltig, M.-O., & Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature 402*(6761), 529–533.

Douglas, R. J., & Martin, K. A. C. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci. 27*, 419–451.

Eppler, J. M., Helias, M., Muller, E., Diesmann, M., & Gewaltig, M. (2009). PyNEST: a convenient interface to the NEST simulator. *Front. Neuroinformatics 2*, 12.

Farquhar, E., & Hasler, P. (2005). A bio-physically inspired silicon neuron. *IEEE Trans. Circuits Syst. I, Reg. Papers 52*(3), 477–488.

Fieres, J. and. Grübl, A., Philipp, S., Meier, K., Schemmel, J., & Schürmann, F. (2004). A platform for parallel operation of VLSI neural networks. In *Proceedings of the 2004 Brain Inspired Cognitive Systems Conference (BICS)*, University of Stirling, Scotland, UK.

Furber, S., Lester, D., Plana, L., Garside, J., Painkras, E., Temple, S., & Brown, A. (2012). Overview of the spinnaker system architecture. *IEEE Trans. Comp. PP*(99), 1.

Gao, P., Benjamin, B., & Boahen, K. (2012). Dynamical system guided mapping of quantitative neuronal models onto neuromorphic hardware. *IEEE Trans. Circuits Syst. I, Reg. Papers 59*(10), 2383–2394.

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia 2*(4), 1430.

Grassia, F., Buhry, L., Lévi, T., Tomas, J., Destexhe, A., & Saïghi, S. (2011). Tunable neuromimetic integrated system for emulating cortical neuron models. *Front. Neurosci. 5*(134).

Grossberg, S. (1973). Contour enhancement, short term memory, and constancies in reverberating neural networks. *Stud. Appl. Math. 52*(3), 213–257.

Gupta, A., Wang, Y., & Markram, H. (2000). Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science 287*, 273–278.

Gütig, R., & Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing-based decisions. *Nat. Neurosci. 9*(3), 420–428.

Häusler, C., Nawrot, M. P., & Schmuker, M. (2011). A spiking neuron classifier network with a deep architecture inspired by the olfactory system of the honeybee. In *2011 5th International IEEE/EMBS Conference on Neural Engineering*, Cancun, Mexico, pp. 198–202. IEEE Press.

Hines, M., Davison, A. P., & Muller, E. (2009). NEURON and Python. *Front. Neuroinformatics 3*.

Imam, N., Akopyan, F., Arthur, J., Merolla, P., Manohar, R., & Modha, D. (2012a). A digital neurosynaptic core using event-driven QDI circuits. In *IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, Lyngby, Denmark, pp. 25–32. IEEE Press.

Imam, N., Cleland, T. A., Manohar, R., Merolla, P. A., Arthur, J. V., Akopyan, F., & Modha, D. S. (2012b). Implementation of olfactory bulb glomerular layer computations in a digital neurosynaptic core. *Front. Neurosci. 6*(83).

Indiveri, G., Chicca, E., & Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw. 17*(1), 211–221.

Indiveri, G., Chicca, E., & Douglas, R. (2009). Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition. *Cognitive Computation 1*(2), 119–127.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., & Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci. 5*(73).

Itti, L., & Koch, C. (2001). Computational modeling of visual attention. *Nat. Rev. Neurosci. 2*(3), 194–203.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, St. Augustin, Germany.

Kaplan, B., Brüderle, D., Schemmel, J., & Meier, K. (2009). High-conductance states on a neuromorphic hardware system. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN)*, Atlanta, pp. 1524–1530. IEEE Press.

Kremkow, J., Aertsen, A., & Kumar, A. (2010). Gating of signal propagation in spiking neural networks by balanced and correlated excitation and inhibition. *J. Neurosci. 30*(47), 15760–15768.

Kremkow, J., Perrinet, L. U., Masson, G. S., & Aertsen, A. (2010). Functional consequences of correlated excitatory and inhibitory conductances in cortical networks. *J. Comput. Neurosci. 28*(3), 579–594.

Kumar, A., Schrader, S., Aertsen, A., & Rotter, S. (2008). The high-conductance state of cortical networks. *Neural Comput. 20*(1), 1–43.

Lazzaro, J., Ryckebusch, S., Mahowald, M., & Mead, C. (1988). Winner-take-all networks of $O(N)$ complexity. In *Advances in Neural Information Processing Systems (NIPS)*, Denver, pp. 703–711. Morgan Kaufmann Publishers.

Linster, C., & Smith, B. H. (1997). A computational model of the response of honey bee antennal lobe circuitry to odor mixtures: overshadowing, blocking and unblocking can arise from lateral inhibition. *Behav. Brain. Res. 87*, 1–14.

Litvak, V., Sompolinsky, H., Segev, I., & Abeles, M. (2003). On the transmission of rate code in long feed-forward networks with excitatory-inhibitory balance. *J. Neurosci. 23*(7), 3006–3015.

Lundqvist, M., Compte, A., & Lansner, A. (2010). Bistable, irregular firing and population oscillations in a modular attractor memory network. *PLoS Comput. Biol. 6*(6), e1000803.

Lundqvist, M., Rehn, M., Djurfeldt, M., & Lansner, A. (2006). Attractor dynamics in a modular network model of neocortex. *Network: Comput. Neural Systems 17*(3), 253–276.

Maass, W. (2000). On the computational power of winner-take-all. *Neural Comput. 12*(11), 2519–2535.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: a new framwork for neural compuation based on perturbation. *Neural Comput. 14*(11), 2531–2560.

Marder, E., & Goaillard, J.-M. (2006). Variability, compensation and homeostasis in neuron and network function. *Nat. Rev. Neurosci. 7*(7), 563–574.

Markram, H., Wang, Y., & Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. *Proc. Natl. Acad. Sci. USA 95*(9), 5323–5328.

Mead, C. (1989). *Analog VLSI and neural systems*. Boston, MA, USA: Addison-Wesley.

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., & Modha, D. (2011). A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In *Proceedings of the 2011 Custom Integrated Circuits Conference (CICC)*, San Jose, USA, pp. 1–4. IEEE Press.

Merolla, P., & Boahen, K. (2006). Dynamic computation in a recurrent network of heterogeneous silicon neurons. In *Proceedings of the 2006 International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, pp. 4539–4542. IEEE Press.

Neftci, E., Chicca, E., Indiveri, G., & Douglas, R. (2011). A systematic method for configuring VLSI networks of spiking neurons. *Neural Comput. 23*(10), 2457–2497.

Neftci, E., & Indiveri, G. (2010). A device mismatch compensation method for VLSI neural networks. In *Proceedings of the 2010 Biomedical Circuits and Systems Conference (BioCAS)*, San Diego, USA, pp. 262–265. IEEE Press.

Nelsen, R. B. (1998). *An Introduction to Copulas (Lecture Notes in Statistics)* (1 ed.). New York, USA: Springer.

Oster, M., Douglas, R., & Liu, S. (2009). Computation with spikes in a winner-take-all network. *Neural Comput. 21*(9), 2437–2465.

Perez-Orive, J., Bazhenov, M., & Laurent, G. (2004). Intrinsic and circuit properties favor coincidence detection for decoding oscillatory input. *J. Neurosci. 24*(26), 6037–47.

Perkel, D. H., Gerstein, G. L., & Moore, G. P. (1967). Neuronal spike trains and stochastic point processes. II. Simultaneous spike trains. *Biophys. J. 7*(4), 419–440.

Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., & Meier, K. (2012). Is a 4-bit synaptic weight resolution enough? - constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci. 6*(90).

Prut, Y., Vaadia, E., Bergman, H., Haalman, I., Hamutal, S., & Abeles, M. (1998). Spatiotemporal structure of cortical activity: Properties and behavioral relevance. *J. Neurophysiol. 79*(6), 2857–2874.

Renaud, S., Tomas, J., Lewis, N., Bornat, Y., Daouzli, A., Rudolph, M., Destexhe, A., & Saïghi, S. (2010). PAX: A mixed hardware/software simulation platform for spiking neural networks. *Neural Networks 23*(7), 905–916.

Rolls, E. T., & Deco, G. (2010). *The Noisy Brain: Stochastic Dynamics as a Principle.* Oxford University Press.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, pp. 1947–1950. IEEE Press.

Schemmel, J., Brüderle, D., Meier, K., & Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated I&F neurons. In *Proceedings of the 2007 International Symposium on Circuits and Systems (ISCAS)*, New Orleans, pp. 3367–3370. IEEE Press.

Schemmel, J., Grübl, A., Meier, K., & Müller, E. (2006). Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, pp. 1–6. IEEE Press.

Schmuker, M., Brüderle, D., Schrader, S., & Nawrot, M. P. (2011). Ten thousand times faster: Classifying multidimensional data on a spiking neuromorphic hardware system. *Front. Comput. Neurosci. Conference Abstract: BC11 : Computational Neuroscience & Neurotechnology Bernstein Conference & Neurex Annual Meeting 2011* (109).

Schmuker, M., & Schneider, G. (2007). Processing and classification of chemical data inspired by insect olfaction. *Proc. Natl. Acad. Sci. USA 104* (51), 20285–20289.

Schrader, S., Diesmann, M., & Morrison, A. (2010). A compositionality machine realized by a hierarchic architecture of synfire chains. *Front. Comput. Neurosci. 4*, 154.

Schrader, S., Grün, S., Diesmann, M., & Gerstein, G. (2008). Detecting synfire chain activity using massively parallel spike train recording. *J. Neurophysiol. 100*, 2165–2176.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodríguez, F., Kolle Riis, H., Delbrück, T., Liu, S.-C., Zahnd, S., Whatley, A. M., Douglas, R., Häfliger, P., Jimenez-Moreno, G., Civit, A., Serrano-Gotarredona, T., Acosta-Jiménez, A., & Linares-Barranco, B. (2006). AER building blocks for multi-layer multi-chip neuromorphic vision systems. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *anips*, pp. 1217–1224. Cambridge, MA: MIT Press.

Soltani, A., & Wang, X.-J. (2010). Synaptic computation underlying probabilistic inference. *Nat. Neurosci. 13* (1), 112–9.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci. 3* (9), 919–926.

Stopfer, M., Bhagavan, S., Smith, B. H., & Laurent, G. (1997). Impaired odour discrimination on desynchronization of odour-encoding neural assemblies. *Nature 390* (6655), 70–4.

Tetzlaff, T., Morrison, A., Timme, M., & Diesmann, M. (2005). Heterogeneity breaks global synchrony in large networks. In *Proceedings of the 30th Göttingen Neurobiology Conference.*

Tsodyks, M. V., & Markram, H. (1997). The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proc. Natl. Acad. Sci. USA 94*, 719–723.

van Vreeswijk, C., & Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science 274*, 1724–1726.

Vogels, T. P., & Abbott, L. F. (2005). Signal propagation and logic gating in networks of integrate-and-fire neurons. *J. Neurosci. 25*(46), 10786–10795.

Vogelstein, R., Mallik, U., Vogelstein, J., & Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw. 18*(1), 253–265.

Wilson, R. I., & Laurent, G. (2005). Role of GABAergic inhibition in shaping odor-evoked spatiotemporal patterns in the drosophila antennal lobe. *J. Neurosci. 25*(40), 9069–9079.

Yazdanbakhsh, A., Babadi, B., Rouhani, S., Arabzadeh, E., & Abbassian, A. (2002). New attractor states for synchronous activity in synfire chains with excitatory and inhibitory coupling. *Biol. Cybern. 86*, 367–378.

II Is a 4-bit synaptic weight resolution enough? – Constraints on enabling spike-timing dependent plasticity in neuromorphic hardware

Thomas Pfeil[1], Tobias C. Potjans[2,3], Sven Schrader,[1], Wiebke Potjans[2,4], Johannes Schemmel[1], Markus Diesmann[2,3,4,5], and Karlheinz Meier[1]

[1] *Kirchhoff Institute for Physics, Ruprecht-Karls-University Heidelberg, Heidelberg, Germany*
[2] *Institute of Neuroscience and Medicine, Computational and Systems Neuroscience (INM-6), Research Center Jülich, Jülich, Germany*
[3] *Brain and Neural Systems Team, RIKEN Computational Science Research Program, Wako-shi, Japan*
[4] *RIKEN Brain Science Institute, Wako-shi, Japan*
[5] *Medical Faculty, RWTH Aachen University, Aachen, Germany*

Abstract

Large-scale neuromorphic hardware systems typically bear the trade-off between detail level and required chip resources. Especially when implementing spike-timing-dependent plasticity, reduction in resources leads to limitations as compared to floating point precision. By design, a natural modification that saves resources would be reducing synaptic weight resolution. In this study, we give an estimate for the impact of synaptic weight discretization on different levels, ranging from random walks of individual weights to computer simulations of spiking neural networks. The FACETS wafer-scale hardware system offers a 4-bit resolution of synaptic weights, which is shown to be sufficient within the scope of our network benchmark. Our findings indicate that increasing the resolution may not even be useful in light of further restrictions of customized mixed-signal synapses. In addition, variations due to production imperfections are investigated and shown to be uncritical in the context of the presented study. Our results represent a general framework for setting up and configuring hardware-constrained synapses. We suggest how weight discretization could be considered for other backends dedicated to large-scale simulations. Thus, our proposition of a *good hardware verification practice* may rise synergy effects between hardware developers and neuroscientists.

1 Introduction

Computer simulations have become an important tool to study cortical networks (e.g. Brunel, 2000; Morrison et al., 2005; Vogels et al., 2005; Markram, 2006; Brette et al., 2007; Johansson & Lansner, 2007; Morrison et al., 2007; Kunkel et al., 2011; Yger et al., 2011). While they provide insight into activity dynamics that can not otherwise be measured *in vivo* or calculated analytically, their computation times can be very time-consuming and consequently unsuitable for statistical analyses, especially for learning neural networks (Morrison et al., 2007). Even the ongoing enhancement of the von Neumann computer architecture is not likely to reduce simulation runtime significantly, as both single- and multi-core scaling face their limits in terms of transistor size (Thompson & Parthasarathy, 2006), energy consumption (Esmaeilzadeh et al., 2011), or communication (Perrin, 2011).

Neuromorphic hardware systems are an alternative to von Neumann computers that alleviates these limitations. Their underlying VLSI microcircuits are especially designed to solve neuron dynamics and can be highly accelerated compared to biological time (Indiveri et al., 2011). For most neuron models whose dynamics can be analytically stated, the evaluation of its equations can be determined either digitally (Plana et al., 2007) by means of numerical methods or with analog circuits that solve the neuron equations intrinsically (Millner et al., 2010). The analog approach has the advantage of maximal parallelism, as all neuron circuits are evolving simultaneously in continuous time. Furthermore, high acceleration factors compared to biological time (e.g. up to 10^5 reported by Millner et al. (2010)), can be achieved by reducing the size of the analog neuron circuits. Nevertheless, many neuromorphic hardware systems are developed for operation in real-time to be applied in sensor applications or medical implants (Fromherz, 2002; Levi et al., 2008; Vogelstein et al., 2008).

Typically, the large number of programmable and possibly plastic synapses accounts for the major part of chip resources in neuromorphic hardware systems (Figure 1). Hence, the limited chip area requires a trade-off between the number and size of neurons and their synapses, while providing sufficiently complex dynamics. For example, decreasing the resolution of synaptic weights offers an opportunity to reduce the area required for synapses and therefore allows more synapses on a chip, rendering the synaptic weights discretized.

In this study, we will analyze the consequences of such a weight discretization and propose generic configuration strategies for spike-timing dependent plasticity on discrete weights. Deviations from original models caused by this discretization are quantified by particular benchmarks. In addition, we will investigate further hardware restrictions specific for the *FACETS*[1] *wafer-scale hardware system* (FACETS, 2010), a pioneering neuromorphic device that implements a large amount of both configurable and plastic synapses (Schemmel et al., 2008, 2010; Brüderle et al., 2011). To this end, custom hardware-inspired synapse models are integrated into a network benchmark using the simulation tool NEST (Gewaltig & Diesmann, 2007). The objective is to determine the smallest hardware implementation of synapses without distorting the behavior of theoretical network models that have been approved by computer simulations.

[1] Fast Analog Computing with Emergent Transient States

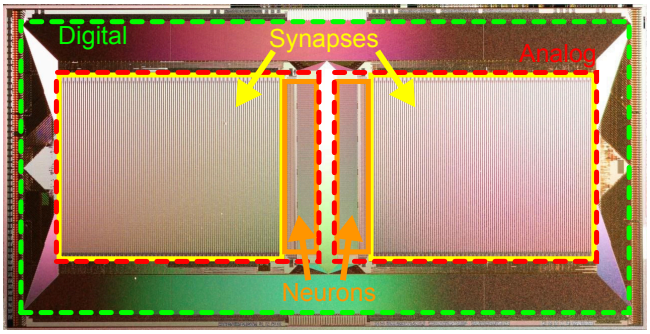


Figure 1: Photograph of the HICANN (High Input Count Analog Neural Network) chip, the basic building block of the FACETS wafer-scale hardware system. Notice the large area occupied by mixed-signal synapse circuits (yellow boxes) compared to neuron circuits (orange boxes). A digital communication infrastructure (area between red and green boxes) ensures a high density of connections between neurons on the same and to other HICANN chips.

2 Materials and Methods

2.1 Spike-timing dependent plasticity

Here, Spike-Timing Dependent Plasticity (STDP) is treated as a pair-based update rule as reviewed by e.g. Morrison et al. (2008). Most pair-based STDP models (Song et al., 2000; van Rossum et al., 2000; Gütig et al., 2003; Morrison et al., 2007) separate weight modifications δw into a spike-timing dependent factor $x(\Delta t)$ and a weight-dependent factor $F(w)$:

$$\delta w(w, \Delta t) = F(w)x(\Delta t), \tag{1}$$

where $\Delta t = t_\mathrm{i} - t_\mathrm{j}$ denotes the interval between spike times t_j and t_i at the pre- and post-synaptic terminal, respectively. Typically, $x(\Delta t)$ is chosen to be exponentially decaying (e.g. Gerstner et al., 1996; Kempter et al., 1999).

In contrast, the weight-dependence $F(w)$, which is divided into $F_+(w)$ for a causal and $F_-(w)$ for an anti-causal spike-timing-dependence, differs between different STDP models. Examples are given in Table 1. As $F_+(w)$ is positive and $F_-(w)$ negative for all these STDP models, causal relationships ($\Delta t > 0$) between pre- and postsynaptic spikes potentiate and anti-causal relationships ($\Delta t < 0$) depress synaptic weights.

In this study, the *intermediate Gütig STDP model* (bounded to the weight range [0,1]) is chosen as an example STDP model. It represents a mixture of the multiplicative ($\mu = 1$) and additive ($\mu = 0$) STDP model and has been shown to provide stability in competitive synaptic learning (Gütig et al., 2003). Nevertheless, the following studies can be applied to any pair-based STDP model with exponentially decaying time-dependence, e.g. all models listed in Table 1.

2.2 Synapses in large-scale hardware systems

The FACETS wafer-scale hardware system (Schemmel et al., 2008, 2010; Brüderle et al., 2011) represents an example for a possible synapse size reduction in neuromorphic hardware systems. Figure 2 schematizes the hardware implementation of a synapse enabling STDP similar as presented in Schemmel et al. (2006) and Schemmel et al. (2007). It

Model name	$F_+(w)$	$F_-(w)$	$x(\Delta t)$
Additive (Song et al., 2000)	λ	$-\lambda\alpha$	$\exp(-\frac{\lvert\Delta t\rvert}{\tau_{\mathrm{STDP}}})$
Multiplicative (Turrigiano et al., 1998)	$\lambda(1-w)$	$-\lambda\alpha w$	
Gütig (Gütig et al., 2003)	$\lambda(1-w)^\mu$	$-\lambda\alpha w^\mu$	
Van Rossum (van Rossum et al., 2000)	c_p	$-c_\mathrm{d} w$	
Power law (Morrison et al., 2007)	λw^μ	$-\lambda\alpha w$	

Table 1: Weight- and spike-timing-dependence of pair-based STDP models: additive, multiplicative, Gütig, van Rossum and power law model. F_+ in case of a causal spike-timing-dependence ($\Delta t > 0$) and F_- in the anti-causal case ($\Delta t < 0$). Throughout this study, the model proposed by Gütig et al. is applied with parameters $\alpha = 1.05$, $\lambda = 0.005$, $\mu = 0.4$ and $\tau_{\mathrm{STDP}} = 20\,\mathrm{ms}$ in accordance with Song et al. (2000); van Rossum et al. (2000); Rubin et al. (2001); Gütig et al. (2003); Morrison et al. (2008).

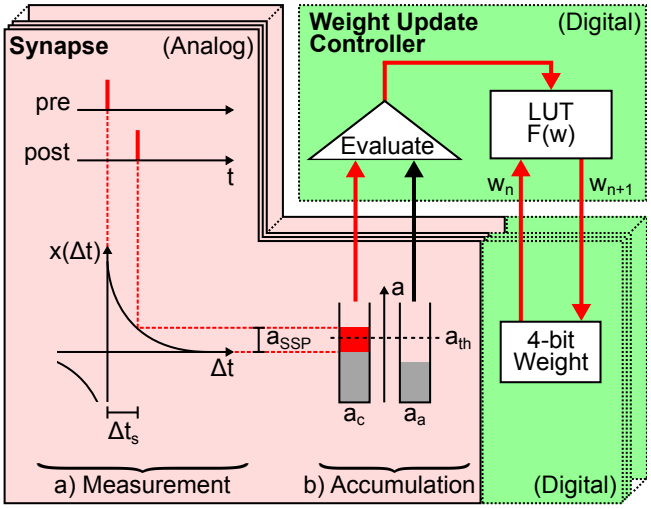


Figure 2: Schematic drawing of local hardware synapses which are consecutively processed by a global weight update controller. Analog circuits are highlighted in red (with solid frame) and digital circuits in green (dashed frames). The spike-timing-dependence (here one standard spike pair (SSP) with Δt_s, see text) between the pre- and postsynaptic neuron is **a)** measured (here a_SSP) and **b)** accumulated (here to a_c in case of a causal spike pair, a_a for anti-causal spike pairs is not affected). Then, the global weight update controller evaluates the accumulated spike-timing-dependence by means of a crossed threshold a_th (here $a_\mathrm{c} > a_\mathrm{th}$) and modifies the digital weight of the hardware synapse accordingly. The new synaptic weight $w_{\mathrm{n}+1}$ is retrieved from the LUT according to the accumulated spike-timing-dependence and the current weight w_n and is written back to the hardware synapse. If either the causal or anti-causal accumulated spike-timing-dependence crosses the threshold, both accumulations are reset to zero. The analog measurement and accumulation circuit is furthermore minimized by using the reduced symmetric nearest-neighbor spike pairing scheme (Morrison et al., 2008): instead of considering all past and future spikes (all-to-all spike pairing scheme), only the latest and the following spike at both terminals of the synapse are taken into account.

provides the functionality to store the value of the synaptic weight, to measure the spike-timing-dependence between pre- and postsynaptic spikes and to update the synaptic weight according to this measurement. Synapse density is maximized by separating the *accumulation* of the spike-timing-dependence $x(\Delta t)$ and the *weight update controller*, which is the hardware implementation of $F(w)$. This allows $4 \cdot 10^7$ synapses on a single wafer (Schemmel et al., 2010).

Synaptic dynamics in the FACETS wafer-scale hardware system exploits the fact that weight dynamics typically evolves slower than electrical neuronal activity (Morrison et al., 2007; Kunkel et al., 2011). Therefore, weight updates can be divided into two steps (Figure 2). First, a measuring and accumulation step which locally determines the relative spike times between pairs of neurons and thus $x(\Delta t)$. This stage is designed in analog hardware (red area in Figure 2), as analog measurement and accumulation circuits require less chip resources compared to digital realizations thereof. Second, the digital weight update controller (upper green area in Figure 2) implements $F(w)$ based on the previous analog result. A global weight update controller[2] is responsible for the consecutive updates of many synapses (Schemmel et al., 2006) and hence limits the maximal rate at which a synapse can be updated, the update controller frequency ν_c.

Sharing one weight update controller reduces synapses to small analog measurement and accumulation circuits as well as a digital circuit that implements the synaptic weight (Figure 2). The area required to implement these digital weights with a resolution of r bits is proportional to 2^r, the number of discrete weights. Consequently, assuming the analog circuits to be fixed in size, the size of a synapse is determined by its weight storage exponentially growing with the weight resolution. E.g. the FACETS wafer-scale hardware system has a weight resolution of $r = 4\,\text{bits}$, letting the previously described circuits (analog and digital) equally sized on the chip.

Modifications in the layout of synapse circuits are time-consuming and involve expensive re-manufacturing of chips. Thus, the configuration of connections between neurons is designed flexible enough to avoid these modifications and provide a general-purpose modeling environment (Schemmel et al., 2010). For the same reason, STDP is conform to the majority of available update rules. The STDP models listed in Table 1 share the same time-dependence $x(\Delta t)$. Its exponential shape is mimicked by small analog circuit not allowing for other time-dependencies (Schemmel et al., 2006, 2007). The widely differing weight-dependences $F(w)$, on the other hand, are programmable into the weight update controller. Due to limited weight update controller resources, arithmetic operations $F(w)$ as listed in Table 1 are not realizable and are replaced by a programmable look-up table (LUT) (Schemmel et al., 2006).

Such a LUT lists, for each discrete weight, the resulting weights in case of causal or anti-causal spike-timing-dependence between pre- and postsynaptic spikes. Instead of performing arithmetic operations during each weight update (Equation 1), LUTs are used as a recallable memory consisting of precalculated weight modifications. Hence, LUTs do not limit the flexibility of weight updates if their weight-dependence (Table 1) does not change over time. Throughout this study, we prefer the concept of LUTs to arithmetic operations, because we like to focus on the discretized weight space, a state space of limited dimension.

In addition to STDP, the FACETS wafer-scale hardware system also supports a variant of short-term plasticity mechanisms according to Tsodyks & Markram (1997) (Schemmel et al., 2007; Bill et al., 2010), which however leaves synaptic weights unchanged and therefore lies outside the scope of this study.

[2]One weight update controller for all 256 neurons with 224 synapses each.

2.3 Discretization of synaptic weights

Continuous weight values $w_c \in [0, 1]$, as assumed for the STDP models listed in Table 1, are transformed into r-bit coded discrete weight values w_d:

$$w_d = c \left\lfloor \frac{w_c}{c} + \frac{1}{2} \right\rfloor \quad \text{for } w_c \in I \tag{2}$$

where $c = 1/(2^r - 1)$ denotes the width of a bin and $\lfloor x \rfloor$ the floor-function, the largest integer less than or equal to x. This procedure divides the range of weight values $I = [0, 1]$ into 2^r bins. The term $\frac{1}{2}$ allows for a correct discretization of weight values near the borders of I, effectively dividing the width of the ending bins (otherwise, only $w_c = 1$ would be mapped to $w_d = 1$).

2.4 Discretization of spike-timing dependent plasticity

A single weight update, resulting from a pre- and postsynaptic spike, might be too fine grained to be captured by a low weight resolution (Equation 2). Therefore, it is necessary to accumulate the effect of weight updates of several consecutive spike pairs in order to reach the next discrete weight value (Equation 2 and Figure 2). This is equivalent to state that the implementation of the STDP model assumes additive features for ms range intervals. To this end, we define a *standard spike pair* (SSP) as a spike pair with a time interval between a pre- and postsynaptic spike of $\Delta t_s = 10\,\text{ms}$ (in accordance to biological measurements by Markram et al., 1997; Bi & Poo, 1998; Sjöström et al., 2001) in order to provide a standardized measure for the spike-timing-dependence. This time interval is chosen arbitrarily defining the granularity only (fine enough for the weight resolutions of interest) and is valid for both pre-post and post-pre spike pairs, as $x(\Delta t)$ takes its absolute value.

The values for a LUT are constructed as follows. First, the parameters r (weight resolution) and n (number of SSPs consecutively applied for an accumulated weight update) as well as the STDP rule-specific parameters τ_{STDP}, λ, μ, α (Table 1) are chosen. Next, starting with a discrete weight w_d, weight updates $\delta w(w, \Delta t_s)$ specified by Equation 1 are recursively applied n times in continuous weight space using either exclusively $F_+(w)$ or $F_-(w)$. This results in two accumulated weight updates $\Delta w_{+/-}$, one for each weight-dependence $F_{+/-}(w)$. Finally, the resulting weight value in continuous space is according to Equation 2 transformed back to its discrete representation. This process is then carried out for each possible discrete weight value w_d (Table 2). We will further compare different LUTs letting n be a free parameter. In the following a *weight update* refers to Δw, if not specified otherwise.

Although we are focusing on the Gütig STDP model, the updated weight values can in general under- or over-run the allowed weight interval I due to finite weight updates Δw. In this case, the weight is clipped to its minimum or maximum value, respectively.

2.5 Equilibrium weight distributions

We analyze long-term effects of weight discretization by studying the equilibrium weight distribution of a synapse that is subject to Poissonian pre- and postsynaptic firing. Thus, potentiation and depression are equally probable ($p_d = p_p = \frac{1}{2}$). Equilibrium weight distributions in discrete weight space of low resolution (between 2 and 10 bits) are compared to those with high resolution (16 bits) via the mean squared error MSE_{eq}. Consecutive weight updates are performed based on precalculated LUTs.

w_d	w_+	w_-
0	$\frac{1}{3}$	0
$\frac{1}{3}$	$\frac{2}{3}$	0
$\frac{2}{3}$	1	$\frac{1}{3}$
1	1	$\frac{2}{3}$

Table 2: Example look-up table for a weight resolution of $r = 2$ bits and $n = 100$ SSPs. Discrete weight w_d and the resulting weight increments $w_{+/-} = w_\mathrm{d} + \Delta w_{+/-}$ for causal and anti-causal weight dependences.

Equilibrium weight distributions of discrete weights for a given weight resolution of r bits are calculated as follows. First, a LUT for 2^r discrete weights is configured with n SSPs. Initially, all 2^r discrete weight values w_i have the same probability $P_{\mathrm{i},0} = \frac{1}{2^r}$. For a compact description, the discrete weights w_i are mapped to a 2^r dimensional space with unit vectors $\vec{e}_\mathrm{i} \in \mathbb{N}^{2^r}$. Then, for each iteration cycle j, the probability distribution is defined by $\vec{P}_\mathrm{j} = \sum_{\mathrm{i}=0}^{2^r-1} P_{\mathrm{i},\mathrm{j}-1}(p_\mathrm{p}\vec{e}_\mathrm{c} + p_\mathrm{d}\vec{e}_\mathrm{a})$, where $P_{\mathrm{i},\mathrm{j}-1}$ is the probability for each discrete weight value w_i of the previous iteration cycle $j-1$. The indices of $\vec{e}_\mathrm{c}$ and $\vec{e}_\mathrm{a}$ are those of the resulting discrete weight values w_i in case of a causal and anti-causal weight update, respectively, and are represented by the LUT. We define an equilibrium state as reached if the Euclidean norm $\left\|\vec{P}_{\mathrm{j}-1} - \vec{P}_\mathrm{j}\right\|$ is smaller than a threshold $h = 10^{-12}$.

An analytical approach for obtaining equilibrium weight distributions is derived in Section 6.1.

2.6 Spiking network benchmarks

In addition to the behavior under Poissonian noise, we study the impact of discretized weights with a software implementation of hardware synapses, enabling us to analyze synapses in isolation as well as in network benchmarks. The design of our simulation environment is flexible enough to take further hardware constraints and biological applications into account.

2.6.1 Software implementation of hardware synapses

The hardware constraints considered in this study are implemented as a customized synapse model within the framework of the NEST simulation tool (Gewaltig & Diesmann, 2007), allowing their well controlled application in simulator-based studies on large-scale neural networks. The basic properties of such a *hardware-inspired synapse model* are described as follows and are illustrated in Figure 2 and Figure 5.

For each LUT configuration defined by its weight resolution r and number n of SSPs, the threshold for allowing weight updates is set to

$$a_\mathrm{th} = n \cdot a_\mathrm{SSP}, \tag{3}$$

defining $a = \sum_\mathrm{i} x(\Delta t_\mathrm{i})$ as the *spike pair accumulation* for arbitrary intervals. Here, a single SSP is used, setting $a = a_\mathrm{SSP} = x(\Delta t_\mathrm{s})$. If either the causal or anti-causal spike pair accumulation $a_\mathrm{c/a}$ crosses the threshold a_th, the synapse is "tagged" for a weight update. At the next cycle of the weight update controller all tagged synapses are updated according to the LUT. Afterwards, the spike pair accumulation (causal or anti-causal) is reset to zero. Untagged synapses remain unprocessed by the update controller, and

spike pairs are further accumulated without performing any weight update. If a synapse accumulates a_c and a_a above threshold between two cycles of the weight update controller, both are reset to zero without updating the synaptic weight.

This threshold process implies that the frequency ν_w of weight updates is dependent on n, which in turn determines the threshold a_{th}, but also on the firing rates and the correlation between the pre- and postsynaptic spike train. In general, a increases faster with higher firing rates or higher correlations. To circumvent these dependencies on network dynamics, we will use n as a generalized description for the weight update frequency ν_w. The weight update frequency ν_w should not be confused with the update controller frequency ν_c, with which is checked for threshold crossings and hence limits ν_w.

Furthermore, we have implemented a *reference synapse model* in NEST, which is based on Gütig et al. (2003). It has the reduction of employing nearest-neighbor instead of all-to-all spike pairing (Morrison et al., 2008).

All simulations involving synapses are simulated with NEST. Spike trains are applied to built-in *parrot neurons*, that simply repeat their input, in order to control pre- and postsynaptic spike trains to interconnecting synapses.

2.6.2 Single synapse benchmark

We compare the weight evolutions of hardware-inspired and reference synapses receiving correlated pre- and postsynaptic spike trains, drawn from a multiple interaction process (MIP) (Kuhn et al., 2003). This process introduces excess synchrony between two realizations by randomly thinning a template Poisson process. SSPs are then obtained by shifting one of the processes by Δt_s.

In this first scenario the spike pair accumulation a is checked for crossing a_{th} with a frequency of $\nu_c = 10\,\text{kHz}$ to focus on the effects of discrete weights only. This frequency is equal to the simulation step size, preventing the spike pair accumulation from overshooting the threshold a_{th} without eliciting a weight update.

Synaptic weights are recorded in time steps of 3 s for an overall period of 150 s and are averaged over 30 random MIP realizations. Afterwards the mean weight at each recorded time step is compared between the hardware-inspired and the reference synapse model by applying the mean squared error MSE_w.

2.6.3 Network benchmarks

The detection of presynaptic synchrony is taken as a benchmark for synapse implementations. Two populations of 10 neurons each converge to an integrate-and-fire neuron with exponentially decaying synaptic conductances (see schematic in Figure 7A and model description in Table 7 and 8) by either hardware-inspired or reference synapses. These synapses are excitatory, and their initial weights are drawn randomly from a uniform distribution over $[0, 1)$. The amplitude of the postsynaptic conductance is wg_{max} with $g_{max} = 100\,\text{nS}$. One population draws its spikes from a MIP with correlation coefficient c (Kuhn et al., 2003), the other from a Poisson process (MIP with $c \to 0$). We choose presynaptic firing rates of 7.2 Hz such that the target neuron settles at a firing rate of $2 - 22\,\text{Hz}$ depending on the synapse model . The exact postsynaptic firing rate is of minor importance as long as the synaptic weights reach an equilibrium state. The synaptic weights are recorded for $2,000\,\text{s}$ with a sampling frequency of 0.1 Hz. The two resulting weight distributions are compared applying the Mann-Whitney U test Mann & Whitney (1947).

Further constraints Not only the discretization of synaptic weights, but also the update controller frequency ν_c and the reset behavior are constraints of the FACETS wafer-scale hardware system.

To study effects caused by a limited update controller frequency, we choose ν_c such that the interval between sequent cycles is a multiple of the simulator time step. Consequently weight updates can only occur on a time grid.

A *common reset* means that both the causal and anti-causal spike pair accumulations are reset, although only either a_c or a_a has crossed a_{th}. Because the common reset requires only one reset line instead of two, it decreases the chip resources of synapses and is implemented in the current FACETS wafer-scale hardware system.

As a basis for a possible compensation mechanism for the common reset, we suggest analog-to-digital converters (ADCs) with a 4-bit resolution that read out the spike pair accumulations. Such ADCs require only a small chip area in the global weight update controller compared to the large area occupied by additional reset lines covering all synapses and are therefore resource saving alternatives to second reset lines. An ADC allows to compare the spike pair accumulations against multiple thresholds. Implementations of the common reset as well as ADCs are added to the existing software model. For multiple thresholds, the same number of LUTs is needed that have to be chosen carefully. To provide symmetry within the order of consecutive causal and anti-causal weight updates, the spike pair accumulation (causal or anti-causal) that dominates in means of crossing a higher threshold is applied first.

Peri-stimulus-time-histograms The difference between static and STDP synapses on eliciting postsynaptic spikes in the above network benchmark can be analyzed with peri-stimulus-time-histograms (PSTHs). Here, PSTHs show the probability of postsynaptic spike occurrences in dependence on the delay between a presynaptic trigger and its following postsynaptic spike. Spike times are recorded within the last third of an elongated simulation of $3,000\,\mathrm{s}$ with $c = 0.025$. During the last $1,000\,\mathrm{s}$ the mean weights are already in their equilibrium state, but are still fluctuating around it. The first spike of any two presynaptic spikes within a time window of $\Delta t_{on} = 1\,\mathrm{ms}$ is used as a trigger. The length of Δt_{on} is chosen small compared to the membrane time constant $\tau_m = 15\,\mathrm{ms}$, such that the excitatory postsynaptic potentials of both presynaptic spikes overlap each other and increase the probability of eliciting a postsynaptic spike. On the other hand Δt_{on} is chosen large enough to not only include the simultaneous spikes generated by the MIP, but also include coincident spikes within the uncorrelated presynaptic population.

2.7 Hardware variations

In contrast to arithmetic operations in software models, analog circuits vary due to the manufacturing process, although they are identically designed. The choice of precision for all building blocks should be governed by those that distort network functionality most. In this study, we assume that variations within the analog measurement and accumulation circuits are likely to be a key requirement for these choices, as they operate on the lowest level of STDP. Circuit variations are measured and compared between the causal and anti-causal part within a synapse and between synapses. All measurements are carried out with the FACETS chip-based hardware system (Schemmel et al., 2006, 2007) with hardware parameters listed in Table 6. The FACETS chip-based hardware system shares a conceptually nearly identical STDP circuit with the FACETS wafer-scale hardware system (for details see Section 6.2) which was still in the assembly process at the course of this

study. The hardware measurements are written in PyNN (Davison et al., 2009) and use the workflow described in Brüderle et al. (2011).

2.7.1 Measurement

The circuit variations due to production imperfection are measured by recording *STDP curves* and comparing their integrals for $\Delta t > 0$ and $\Delta t < 0$. The curves are recorded by applying equidistant pairs of pre- and postsynaptic spikes with a predefined latency Δt. Presynaptic spikes can be fed into the hardware precisely. However, in contrast to NEST's parrot neurons, postsynaptic spikes are not directly adjustable and therefore have to be evoked by several synchronous external triggers (for details see Section 6.3). After discarding the first 10 spike pairs to ensure regular firing, the pre- and postsynaptic spike trains are shifted until the desired latency Δt is measured. Due to the low spike pair frequency of 10 Hz, only the correlations within and not between the spike pairs are accumulated. The number N of consecutive spike pairs is increased until the threshold is crossed and hence a correlation flag is set (Figure 8A). The inverse of this number versus Δt is called an STDP curve. Such curves were recorded for 252 synapses within one synapse column, the remaining 4 synapses in this column were discarded.

For each STDP curve the total area $A_\mathrm{t} = A_\mathrm{a} + A_\mathrm{c}$ is calculated and normalized by the mean $\overline{A_\mathrm{abs}}$ of the absolute area $A_\mathrm{abs} = |A_\mathrm{a}| + |A_\mathrm{c}|$ over all STDP curves. Ideally, A_t would vanish if both circuits are manufactured identically. The standard deviation σ_a (assuming Gaussian distributed measurement data) of these normalized total areas A_t is taken as one measure for circuit variations. Besides this asymmetry which measures the variation *within* a synapse, a measure for variation *across* synapses is the standard deviation σ_t of the absolute areas A_abs. Therefore the absolute areas A_abs under each STDP curve are again normalized by $\overline{A_\mathrm{abs}}$ and furthermore the mean of all these normalized absolute areas is subtracted.

2.7.2 Software analysis

In order to predict the effects of the previously measured variations on the network benchmark, these variations are integrated into computer simulations. The thresholds for the causal and anti-causal spike pair accumulations are drawn from two overlaying Gaussian distributions defined by the ideal thresholds (Equation 3) and their variations σ_t, σ_a. Again, the same network benchmark as described above is used, but with a fixed correlation coefficient of $c = 0.025$ and an 8-bit LUT configured with $n = 12$ SSPs.

3 Results

Synaptic weights of the FACETS wafer-scale hardware system (Schemmel et al., 2010) have a 4-bit resolution. We show that such a weight resolution is enough to exhibit learning in a neural network benchmark for synchrony detection. To this end, we analyze the effects of weight discretization in three steps as summarized in Table 3.

Description	Results	Methods
Look-up table analysis: Basic analyses on the configuration of STDP on discrete weights by means of look-up tables (A)	A) Section 3.1	A) Section 2.3 and 2.4
and their long-term dynamics (B).	B) Section 3.2	B) Section 2.5
Spiking network benchmarks: Software implementation of hardware-inspired synapses with discrete weights for application in spiking neural environments (C).		C) Section 2.6.1
Analyses of their effects on short-term weight dynamics in single synapses (D)	D) Section 3.3.1	D) Section 2.6.2
and neural networks (E).	E) Section 3.3.2	E) Section 2.6.3
Analyses on how additional hardware constraints effect the network benchmark (F).	F) Section 3.3.3	F) Section 2.6.3
Hardware measurements: Measurement of hardware variations (G)	G) Section 3.4	G) Section 2.7.1
and computer simulations analyzing their effects on the network benchmark (H).	H) Section 3.4	H) Section 2.7.2

Table 3: Outline of analyses on the effects of weight discretization and further hardware constraints.

3.1 Dynamic range of STDP on discrete weights

We choose the configuration of STDP on discrete weights according to Section 2.3 and Section 2.4 to obtain weight dynamics comparable to that in continuous weight space. Each configuration can be described by a LUT "projecting" each discrete weight to new values, one for potentiation and one for depression. For a given weight resolution r the free configuration parameter n (number of SSPs) has to be adjusted to avoid a further reduction of the usable weight resolution by *dead discrete weights*. Dead discrete weights are defined as weights projecting to themselves in case of both potentiation and depression or not receiving any projections from other discrete weights. The percentage of dead discrete weights d defines the lower and upper limit of feasible values for n, the *dynamic range*. The absolute value of the interval within a SSP (Δt_s) is an arbitrary choice merely defining the granularity, but does not affect the results (not shown). Note that spike timing precision in vivo, which is observed for high dimensional input such as dense noise and natural scenes, goes rarely beyond 5 to 10 ms (Butts et al., 2007; Desbordes et al., 2008; Marre et al., 2009; Desbordes et al., 2010; Frégnac, 2012), and the choice of 10 ms as a granular step is thus justified biologically.

Generally, low values of n realize frequent, small weight updates. However, if n is too low, some discrete weights may project to themselves (see rounding in Equation 2) and prevent synaptic weights from evolving dynamically (see Table 4b and $n = 15$ in Figure 3A).

On the other hand, if n exceeds the upper limit of the dynamic range, intermediate discrete weights may not be reached by others. Rare, large weight updates favor projections to discrete weights near the borders of the weight range I and lead to a bimodal equilibrium weight distribution as shown in Table 4c and Figure 3A ($n = 500$).

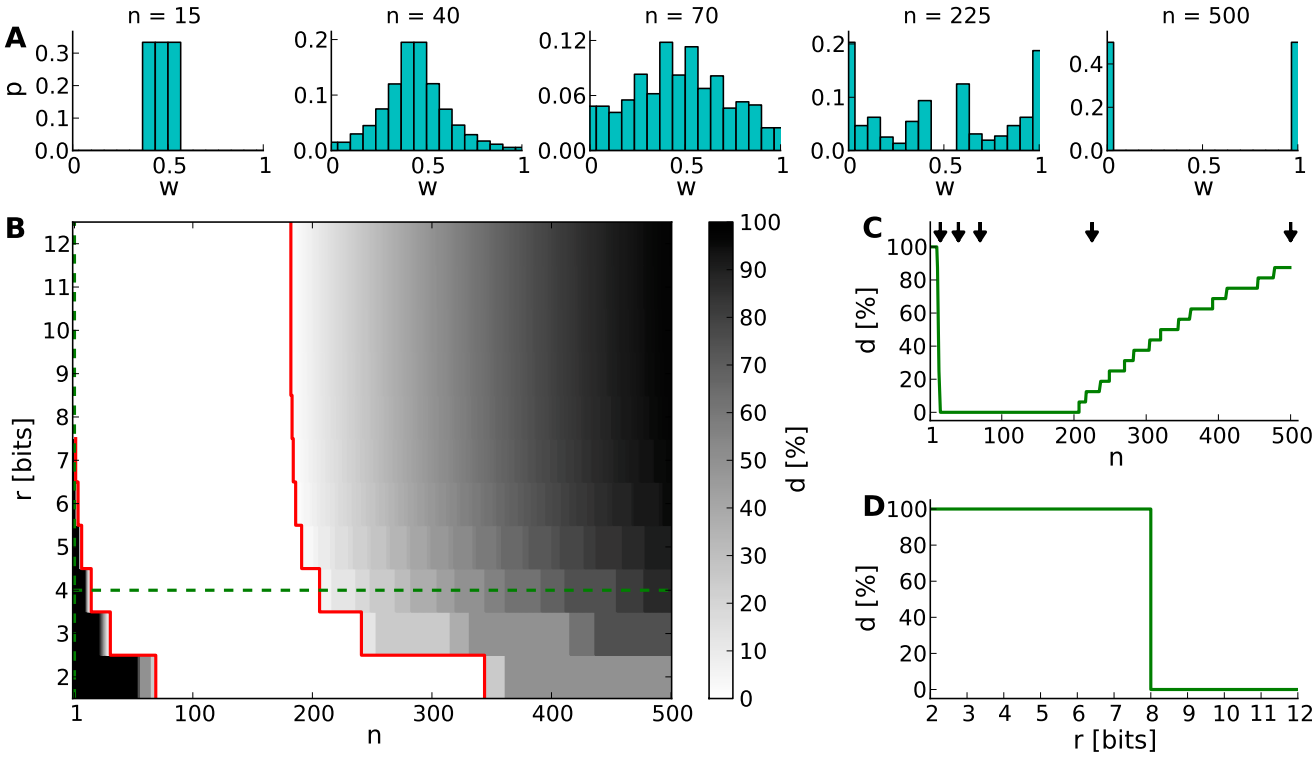


Figure 3: The dynamic range for configurations of STDP on discrete weights. **(A)** Equilibrium weight distributions for a 4-bit weight resolution: Intermediate discrete weights partly project to themselves ($n = 15$). The equilibrium weight distribution widens with an increasing number of SSPs ($n = 40$ and $n = 70$). For a large number of SSPs ($n = 225$ and $n = 500$) the intermediate discrete weights do not receive projections from others. **(B)** Percentage of dead discrete weights d. The limits of the dynamic range ($d = 0\%$) are highlighted in red. The limit towards low numbers of SSPs ($n = 15$ in case of $r = 4\,\text{bits}$) is caused by rounding effects (Equation 2), whereas the upper limit ($n = 206$ in case of $r = 4\,\text{bits}$) is caused by too large weight updates. Green dashed lines indicate cross sections shown in (C) and (D). **(C)** Cross section of (B) at a 4-bit weight resolution. The histograms shown in (A) are depicted with arrows. **(D)** Cross section of (B) at $n = 1$.

w_d	w_+	w_-
0	$\frac{1}{3}$	0
$\frac{1}{3}$	$\frac{2}{3}$	0
$\frac{2}{3}$	1	$\frac{1}{3}$
1	1	$\frac{2}{3}$

(a)

w_d	w_+	w_-
0	$\frac{1}{3}$	0
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$
1	1	$\frac{2}{3}$

(b)

w_d	w_+	w_-
0	$\frac{2}{3}$	0
$\frac{1}{3}$	1	0
$\frac{2}{3}$	1	0
1	1	0

(c)

Table 4: Look-up tables for different numbers n of SSPs. (a) As in Table 2 ($n = 100$), which results in a LUT as expected. Weights are either potentiated or depressed through the entire table. (b) $n = 60$, which is too low, because the discrete weights $\frac{1}{3}$ and $\frac{2}{3}$ are projecting exclusively to themselves. (c) $n = 350$, which is too large, because for w_+ the discrete weight 0 is mapped right to $\frac{2}{3}$ (and for w_- the weight 1 is mapped to 0), thus $\frac{1}{3}$ is never reached.

The lower limit of the dynamic range decreases with increasing resolution (Figure 3B). Compared to a 4-bit weight resolution, an 8-bit weight resolution is sufficiently high to resolve weight updates down to a single SSP (Figure 3D). This allows frequent weight updates comparable to weight evolutions in continuous weight space. The upper limit of the dynamic range does not change over increasing weight resolutions, but is critical for limited update controller frequencies as investigated in Section 3.3.

3.2 Equilibrium weight distributions

Studying learning in neural networks may span long periods of time. Therefore we analyze equilibrium weight distributions being the temporal limit of Poissonian distributed pre- and postsynaptic spiking. These distributions are obtained by applying random walks on LUTs with uniformly distributed occurrences of potentiations and depressions (Section 2.5). Figure 4A shows i.a. boundary effects caused by LUTs configured within the upper part of the dynamic range. E.g. for $n = 144$, the relative frequencies of both boundary values are increased due to large weight steps (red and cyan distributions). Frequent weights, in turn, increase the probability of weights to which they project (according to the LUT). This effect decreases with the number of look-ups, due to the random nature of the stimulus, however, causing intermediate weight values to occur at higher probability.

The impact of weight discretization on long-term weight dynamics is quantified by comparing equilibrium weight distributions between low and high weight resolutions. Weight discretization involves distortions caused by rounding effects for small n (Equation 2 and Figure 3) and boundary effects for high n (Figure 4A and C). High weight resolutions can compensate for rounding effects, but not for boundary effects (Figure 4B).

This analysis on long-term weight dynamics (Figure 4C) refines the choice for n roughly estimated by the dynamic range (Figure 3C).

3.3 Spiking network benchmarks

We extend the above studies on temporal limits by analyses on short-term dynamics with unequal probabilities for potentiation p_p and depression p_d. A hardware-inspired synapse model is used in computer simulations of spiking neural networks, of which an example of typical dynamics is shown in Figure 5. As the pre- and postsynaptic spike trains are correlated in a causal fashion, the causal spike pair accumulation increases faster than the

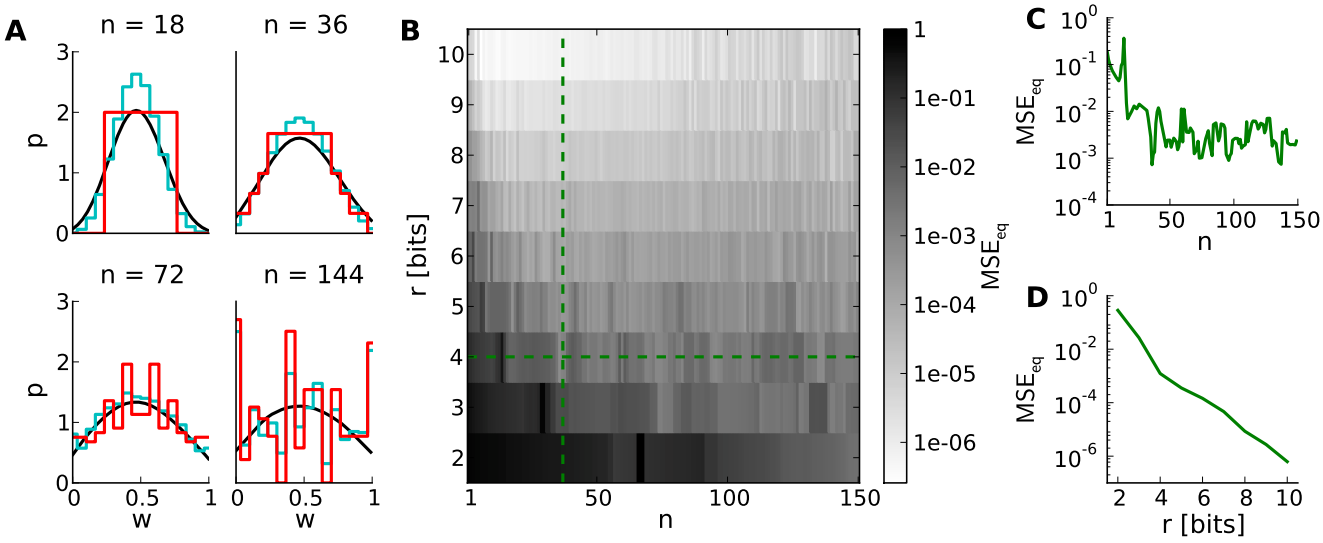


Figure 4: Equilibrium weight distributions (long-term weight evolutions) for configurations of STDP on discrete weights. **(A)** Equilibrium weight distributions for weight resolutions of $r = 4$ bits (red) and $r = 16$ bits (cyan). Both distributions are displayed in 4-bit sampling, for better comparison. Black curves depict the analytical approach. We have chosen $j = 10^5$ iterations for generating each discrete weight distribution to ensure convergence to the equilibrium state. **(B)** Mean squared error MSE_{eq} between the equilibrium weight distributions for weight resolutions r and the reference weight resolution of 16 bits versus the number n of SSPs. **(C),(D)** Cross sections of (B) at $r = 4$ bits and $n = 36$, respectively.

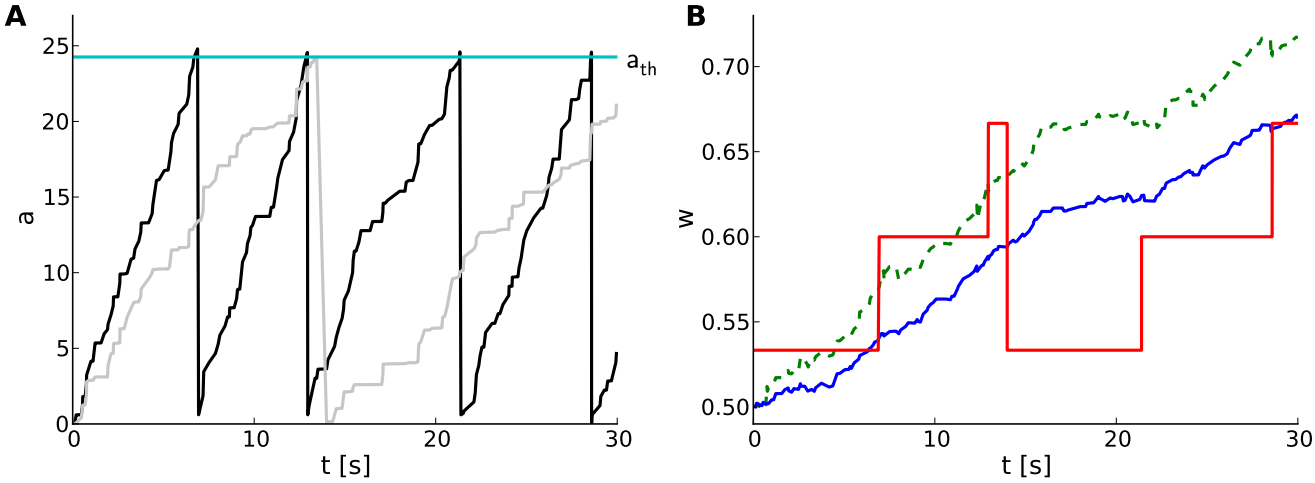


Figure 5: Software implementation of STDP on discrete weights in spiking neural networks. **(A)** Temporal evolution of spike pair accumulations a (dimensionless) for causal (black) and anti-causal (gray) spike-timing-dependences. If a crosses the threshold a_{th} (cyan), the weight is updated and a is reset to zero. Pre- and postsynaptic spike trains are generated by a MIP with $c = 0.5$ and $r = 10\,\text{Hz}$. **(B)** Corresponding weight evolution (solid red) for a 4-bit weight resolution and a LUT configured with $n = 30$. The weight evolution of the reference synapse model with continuous weights, but a reduced symmetric nearest-neighbor spike pairing scheme is depicted in solid blue. It differs from that of a synapse model with continuous weights and an all-to-all spike pairing scheme (dashed green).

anti-causal one (Figure 5A). It crosses the threshold twice, evoking two potentiation steps (at around 7 s and 13 s) before the anti-causal spike pair accumulation evokes a depression at around 14 s (Figure 5A and B). The first two potentiations project to the subsequent entry of the LUT, whereas the following depression rounds to the next but one discrete weight (omitting one entry in the LUT) due to the asymmetry measure α in the STDP model by Gütig et al. (2003).

3.3.1 Single synapse benchmark

This benchmark compares single weight traces between hardware-inspired and reference synapses (Section 2.6.2). A synapse receives correlated pre- and postsynaptic input (Figure 6A) resulting in weight dynamics as shown in Figure 6B. The standard deviation for discrete weights (hardware-inspired synapse model) is larger than that for continuous weights (reference model). This difference is caused by rare, large weight jumps (induced by high n) also responsible for the broadening of equilibrium weight distributions (Figure 4A). Consequently, the standard deviation increases further with decreasing weight resolutions (not shown here).

The dependence of the deviation between discrete and continuous weight traces on the weight resolution r and the number n of SSPs is qualitatively comparable to that of comparisons between equilibrium weight distributions (Figure 6D and E). This similarity, especially in dependence on n (Figure 6D), emphasizes the crucial impact of LUT configurations on both short- and long-term weight dynamics.

To further illustrate underlying rounding effects when configuring LUTs, the asymmetry value α in Gütig's STDP model can be taken as an example. In an extreme case

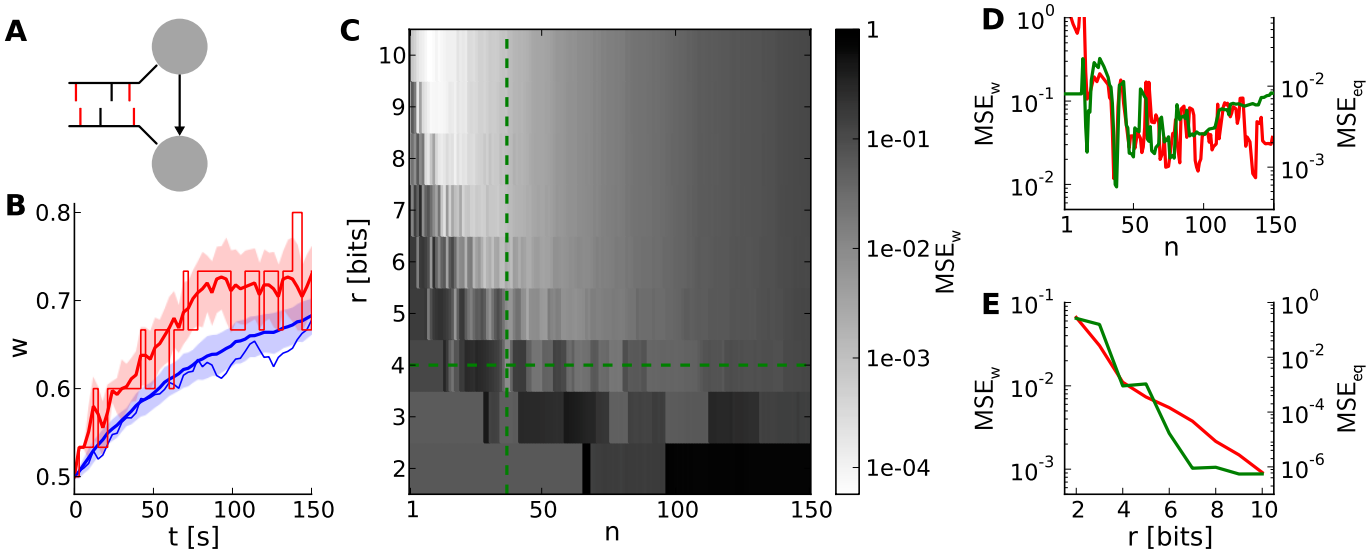


Figure 6: Weight evolution of a single synapse with discrete weights. **(A)** Network layout for single synapse analyses. An STDP synapse (arrow) connects two neurons receiving correlated spike trains with correlation coefficient c (correlated spikes in red bars). **(B)** Example weight traces for the hardware-inspired ($r = 4\,\text{bits}$, $n = 36$ in red) and reference synapse model (blue). Means and standard deviations over 30 realizations are plotted as bold lines and shaded areas, respectively. The single weight traces for one arbitrarily chosen random seed are depicted as thin lines. We applied a correlation coefficient $c = 0.2$, an initial weight $w_0 = 0.5$ and firing rates of 10 Hz. The results persist qualitatively for differing values staying within biologically relevant ranges (not shown here). **(C)** Mean squared error MSE_w between the mean weight traces as shown in (A) over the weight resolution r and the number n of SSPs. The parameters c, w_0 and the firing rates are chosen as in (B). Other values for c and w_0 do not change the results qualitatively. **(D),(E)** Cross sections of (C) at $r = 4\,\text{bits}$ and $n = 36$ in green. Red curves are adapted from Figure 4C and D.

both potentiation and depression are rounded down (compare weight step size for potentiation and depression in Figure 5B). This would increase the originally slight asymmetry drastically and therefore enlarge the distortion caused by weight discretization.

The weight update frequency ν_w is determined by the weight resolution r and the number n of SSPs. High frequencies are beneficial for chronologically keeping up with weight evolutions in continuous weight space. They can be realized by small numbers of SSPs lowering the threshold a_th (Equation 3). On the other hand, rounding effects in the LUT configuration deteriorate for too small numbers of SSPs (Figure 6D). In case of a weight resolution $r = 4\,\text{bits}$ ($r = 8\,\text{bits}$) choosing $n = 36$ ($n = 12$) for the LUT configuration represents a good balance between a high weight update frequency and proper both short- and long-term weight dynamics (Figure 3B, Figure 4B and Figure 6C). Note that n can be chosen smaller for higher weight resolutions, because the distorting impact of rounding effects decreases.

3.3.2 Network benchmark: synchrony detection

Not only exact weight traces of single synapses (Section 3.3.1), but rather those of synapse populations are crucial to fulfill tasks, e.g. the detection of synchronous firing within

neural networks. The principle of synchrony detection is a crucial feature of various neural networks with plasticity, e.g. reported by Senn et al. (1998); Kuba et al. (2002); Davison & Frégnac (2006); El Boustani et al. (2012). Here, it is introduced by means of an elementary benchmark neural network (Figure 7A and Section 2.6.3), using the hardware-inspired or reference synapse model, respectively.

Figure 7B shows a delay distribution of postsynaptic spike occurrences, relative to the trigger onset, synchronous presynaptic firing (Section 2.6.3). For the shown range of Δt_{del}, the postsynaptic neuron is more likely to fire if connected with static (dark gray trace) instead of STDP (black trace) synapses. The correlated population causes its afferent synapses to strengthen more compared to those from the uncorrelated population. This can be seen in Figure 7C, where w saturates at different values ($t \approx 700\,\mathrm{s}$). The same effect can be observed for discretized weights in Figure 7D. For $\Delta t_{\mathrm{del}} > 170\,\mathrm{ms}$ the delay distribution for static synapses is larger than that for STDP synapses (not shown here), because such delayed postsynaptic spikes are barely influenced by their presynaptic counterparts. This is due to small time constants of the postsynaptic neuron (see $\tau_{\mathrm{m}} = \frac{C_m}{g_L}$ and τ_{syn} in Table 7 and 8) compared to Δt_{del}.

Figure 7E shows the p-values of the Mann-Whitney U test applied to both groups of synaptic weights at $t = 2,000\,\mathrm{s}$ for different configurations of weight resolution r and number n of SSPs. Generally, p-values (probability of having the same median within both groups of weights) decrease with an increasing correlation coefficient. Although applying previously selected "healthy" LUT configurations, weight discretization changes the required correlation coefficient for reaching significance level (gray shaded areas). Incrementing the weight resolution while retaining the number of SSPs n does not change the p-values significantly. Low weight resolutions cause larger spacings between discrete weights that can further facilitate the distinction between both medians (for $n = 36$ compare $r = 4\,\mathrm{bits}$ to $r = 8\,\mathrm{bits}$ bits in Figure 7E). However, reducing n for high weight resolutions shortens the accumulation period and consequently allows the synapses to capture fluctuations in a on smaller time scales. This improves the p-value, but is inconvenient for low weight resolutions, because these LUT configurations do not yield the desired weight dynamics (Figure 3, 4 and 6).

3.3.3 Network benchmark: further constraints

In addition to the discretization of synaptic weights that has been analyzed so far, we also consider additional hardware constraints of the FACETS wafer-scale system (Section 2.6.3). This allows us to compare the effects of other hardware constraints to those of weight discretization.

First, we take into account a limited update controller frequency ν_{c}. Figure 7F shows that low frequencies ($< 1\,\mathrm{Hz}$) distort the weight dynamics drastically and deteriorate the distinction between correlated and uncorrelated inputs. Ideally, a weight update would be performed whenever the spike pair accumulations cross the threshold (Figure 5A). However, these weight updates of frequency ν_{w} are now limited to a time grid with frequency ν_{c}. The larger the latency between a threshold crossing and the arrival of the weight update controller, the more likely this threshold is exceeded. Hence, the weight update is underestimated and delayed. Low weight resolutions are less affected, because a high ratio $\frac{\nu_{\mathrm{c}}}{\nu_{\mathrm{w}}}$ reduces threshold overruns and hence distortions. This low resolution requires a high number of SSPs which in turn increases the threshold a_{th} (Equation 3) and thus the weight update frequency ν_{w}.

Second, hardware-inspired synapses with the limitation to common reset lines cease to discriminate between correlated and uncorrelated input (Figure 7G, yellow and magenta

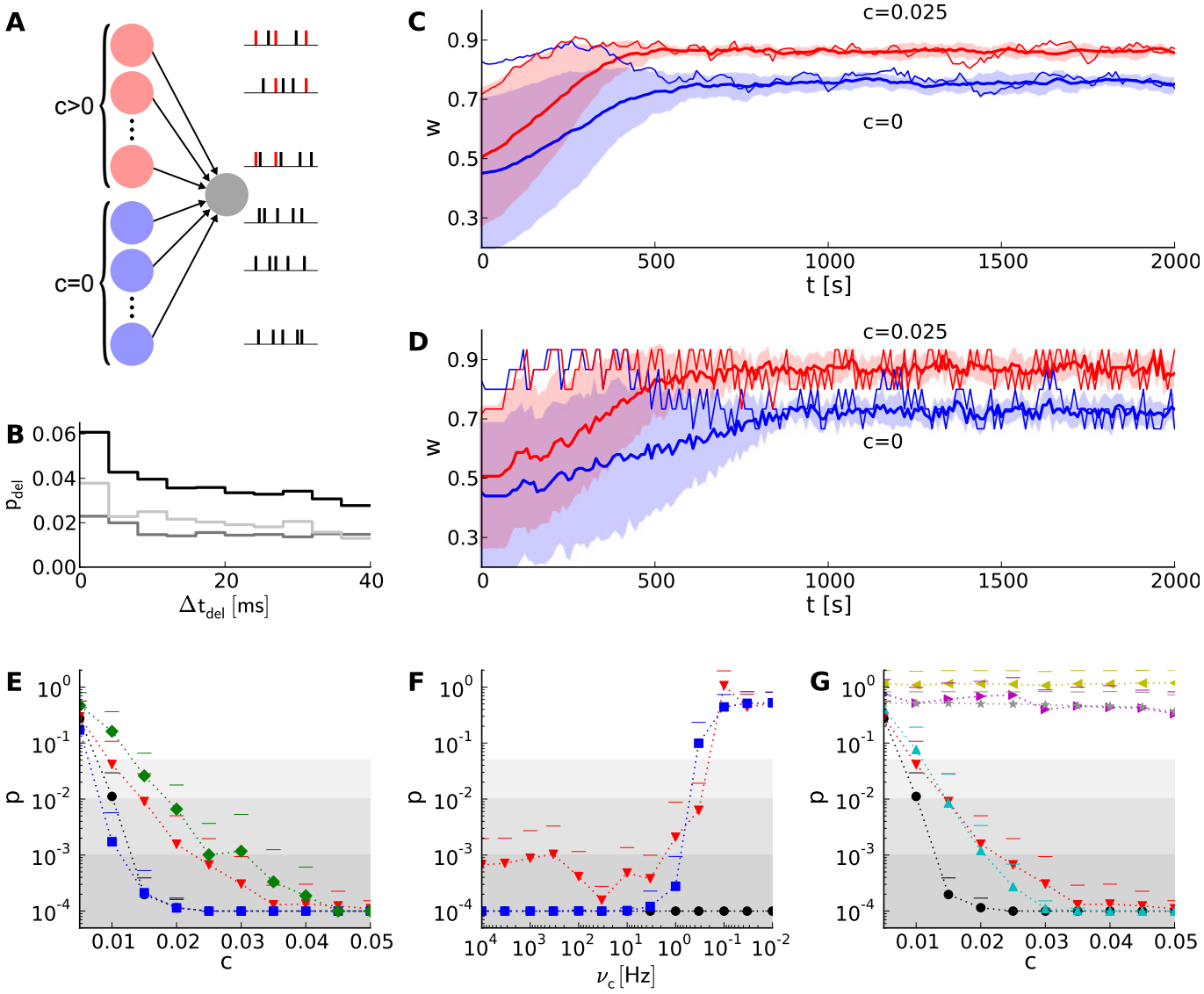


Figure 7: Learning with discrete weights in a neural network benchmark for synchrony detection. **(A)** Layout of the network benchmark. Two populations of presynaptic neurons are connected to a postsynaptic neuron. On the right, example spike trains of the presynaptic neurons are shown. Red spikes indicate correlated firing due to shared spikes. **(B)** PSTH for static synapses and STDP reference synapses. The light gray histogram shows the difference between a simulation with STDP reference synapses (black) and static synapses (dark gray). **(C)** The mean weight traces (thick lines) and their standard deviations (shaded areas) for both populations of afferent synapses using the reference synapses model. Thin lines represent single synapses randomly chosen for each population. **(D)** As in (B), but with the hardware-inspired synapse model ($r = 4\,\text{bits}$ and $n = 36$). **(E)** The probability (p-value of Mann-Whitney U test) of having the same median of weights within both groups of synapses (with correlated and uncorrelated input) at $t = 2,000\,\text{s}$ versus the correlation coefficient c. The hardware-inspired synapse model is represented in red ($r = 4\,\text{bits}$ and $n = 36$), green ($r = 8\,\text{bits}$ and $n = 36$) and blue ($r = 8\,\text{bits}$ and $n = 12$). Black depicts the reference synapse model ($r = 64\,\text{bits}$). The background shading represents the significance levels: $p < 0.05$, $p < 0.01$ and $p < 0.001$. **(F)** Dependence of the p-value on the update controller frequency ν_c for $c = 0.025$. Colors as in (E) **(G)** Black and red trace as in (E). Additionally, p-values for hardware-inspired synapses with common resets are plotted in yellow ($r = 4\,\text{bits}$ and $n = 36$) and magenta ($r = 8\,\text{bits}$ and $n = 12$). Compensations with ADCs are depicted in gray ($r = 4\,\text{bits}$ and $n = 15$ to 45 in steps of 2) and cyan ($r = 8\,\text{bits}$ and $n = 1$ to 46 in steps of 3).

traces). A crossing of the threshold by one spike pair accumulation resets the other (Figure 5) and suppresses its further weight updates, leading to underestimation of synapses with less correlated input.

To compensate for common resets we suggest ADCs that allow the comparison of spike pair accumulations to multiple thresholds. Nevertheless, ADCs compensate common resets only for high weight resolutions (Figure 7G). Again, for low weight resolutions and hence high numbers of SSPs fluctuations can not be taken into account (Figure 7G, gray values). This is the case for a 4-bit weight resolution, whereas a 8-bit weight resolution is high enough to resolve small fluctuations down to single SSPs (Figure 7G, cyan values). Each threshold has its own LUT configured with a number of SSPs that matches the dynamic range (Figure 3). The upper limit of n is chosen according to the results of Section 3.2. The update controller frequency is chosen to be low enough ($\nu_\mathrm{c} = 0.2\,\mathrm{Hz}$) to enable all thresholds to be hit.

3.4 Hardware variations

So far, we neglected production imperfections in real hardware systems. However, fixed pattern noise induced by these imperfections are a crucial limitation on the transistor level and may distort the functionality of the analog synapse circuit making higher weight resolutions unnecessary. The smaller and denser the transistors, the larger the discrepancies from their theoretical properties (Pelgrom et al., 1989). Using the protocol illustrated in Figure 8A we recorded STDP curves on the FACETS chip-based hardware system (Figure 8B, C and Section 2.7.1). Variations within (σ_a) and between (σ_t) individual synapses are shown as distributions in Figure 8D and E, both suggesting variations at around 20%. Both variations are incorporated into computer simulations of the network benchmark (Figure 7A and Section 2.7.2) to analyze their effects on synchrony detection. The p-value (as in Figure 7E-G) rises with increasing asymmetry within synapses, but is hardly affected by variations between synapses (Figure 8F).

4 Discussion

4.1 Configuration of STDP on discrete weights

In this study, we demonstrate generic strategies to configure STDP on discrete weights as e.g. implemented in neuromorphic hardware systems. Resulting weight dynamics is critically dependent on the frequency of weight updates that has to be adjusted to the available weight resolution. Choosing a frequency within the dynamic range (Figure 3) is a prerequisite for the exploitation of discrete weight space ensuring proper weight dynamics. Analyses on long-term dynamics using Poisson-driven equilibrium weight distributions help to refine this choice (Figure 4). The obtained configuration space is similar to that of short-term dynamics, being the evolution of single synaptic weights (Figure 6). This similarity confirms the crucial impact of the LUT configuration on weight dynamics which is caused by rounding effects. Based on these results, we have chosen two example LUT configurations ($r = 4\,\mathrm{bits}$; $n = 36$ and $r = 8\,\mathrm{bits}$; $n = 12$) for further analysis, both realizable on the FACETS wafer-scale hardware system. High weight resolutions allow for higher frequencies of weight updates approximating the ideal model, occasionally requiring several spike pairs to evoke a weight update. Correspondingly, in associative pairing literature, a minimal number of associations is required to detect functional changes (expressed by the spiking or postsynaptic potential response) and varies from studies to studies from a few to several tens (Cassenaer & Laurent, 2007, 2012).

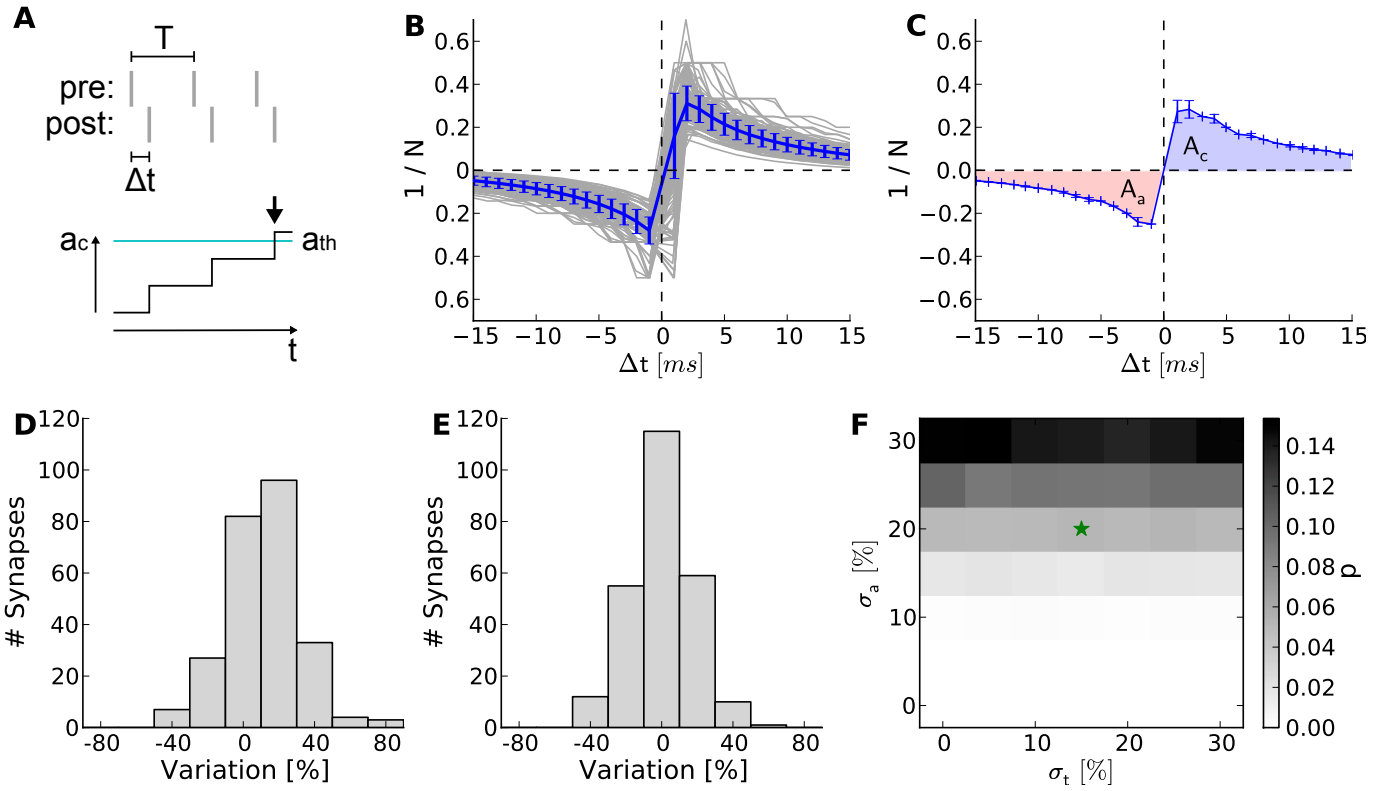


Figure 8: Measurement of hardware synapse variations and their effects on learning in the neural network benchmark. **(A)** Setup for recording STDP curves. At the top, spike trains of the pre- and postsynaptic neuron. Spike pairs with latency Δt are repeated with frequency $\frac{1}{T}$. At the bottom, a spike pair accumulation that crosses the threshold a_{th} (arrow). The inverse of the number of SSPs until crossing a_{th} (here $N = 3$) is plotted in (B). **(B)** STDP curves of 252 hardware synapses within one synapse column (gray) and their mean with error (blue). A speed-up factor of 10^5 is assumed. These curves correspond to $x(\Delta t)$ in Equation 1, whereas $F(w)$ is realized by the LUT. **(C)** One arbitrarily chosen STDP curve (over 5 trials) showing the areas for $\Delta t < 0$ (A_{a} in red) and $\Delta t > 0$ (A_{c} in blue). **(D)** Asymmetry between A_{a} and A_{c} within synapses ($\sigma_{\text{a}} = 21\%$). **(E)** Variation of the absolute areas between synapses ($\sigma_{\text{t}} = 17\%$). **(F)** The p-value (as in Figure 7E-G) in dependence on σ_{a} and σ_{t}. The values for (D) and (E) are marked with an asterisk.

Discretization not only affects the accuracy of weights, but also broadens their equilibrium weight distributions (Figure 4), which are actually shown to be narrow in large-scale neural networks (Morrison et al., 2007). Furthermore, this broadening can distort the functionality of neural networks, e.g. it deteriorates the distinction between the two groups of weights (of synapses originating from the correlated or uncorrelated population) within the network benchmark (compare Figure 7C to D). On the other hand, weight discretization can also be advantageous for synchrony detection, if e.g. groups of weights separate due to large step sizes between neighboring discrete weights (compare red and green in Figure 7E).

In summary, these analyses of STDP on discrete weights are necessary for obtaining appropriate configurations for a variety of STDP models and weight resolutions.

4.2 4-bit weight resolution

Simulations of the network benchmark show that a 4-bit weight resolution is sufficient to detect synchronous presynaptic firing significantly (Figure 7). Groups of synapses receiving correlated input strengthen and in turn increase the probability of synchronous presynaptic activity to elicit postsynaptic spikes as compared to static synapses (Figure 7B). Thus, the weight distribution within the network reflects synchrony within sub-populations of presynaptic neurons. Increasing the weight resolution causes both weight distributions, for the correlated and uncorrelated input, to narrow and separate from each other. Consequently, an 8-bit resolution is sufficient to reproduce the p-values of continuous weights with floating point precision (corresponds to discrete weights with $r = 64\,\text{bits}$, Figure 7E). This resolution requires the combination of two hardware synapses and is under development (Schemmel et al., 2010). On the other hand, increasing the weight resolution, but retaining the frequency of weight updates (number of SSPs), results in weight distributions of comparable width and consequently does not improve the p-values significantly (Figure 7E).

Other neuromorphic hardware systems implement bistable synapses corresponding to a 1-bit weight resolution (Badoni et al., 2006; Indiveri et al., 2010). Bistable synapse models are shown to be sufficient for memory formation (Amit & Fusi, 1994; Fusi et al., 2005; Brader et al., 2007; Clopath et al., 2008). However, these models do not only employ spike timings (Levy & Steward, 1983; Markram et al., 1997; Bi & Poo, 2001; Mu & Poo, 2006; Cassenaer & Laurent, 2007), but also read the postsynaptic membrane potential (Sjöström et al., 2001; Trachtenberg et al., 2002) requiring additional hardware resources. So far, there is no consensus of a general synapse model, and neuromorphic hardware systems are mostly limited to only subclasses of these models.

This studies on weight discretization are not limited to the FACETS hardware systems only, but are applicable to other backends for neural network simulations. For example, our results can be applied to the fully digital neuromorphic hardware system described by Jin et al. (2010b), who also report STDP with a reduced weight resolution. Furthermore, weight discretization may be a further approach to reduce memory consumption of "classical" neural simulators.

4.3 Further hardware constraints

In addition to a limited weight resolution, we have studied further constraints of the current FACETS wafer-scale hardware system with the network benchmark.

A limited update controller frequency implying a minimum time interval between subsequent weight updates does not affect the p-values down to a critical frequency $\nu_c \approx 1\,\text{Hz}$

(Figure 7F). The update controller frequency decreases linearly with the number of hardware synapses enabled for STDP. Assuming a hardware acceleration factor of 10^3 all synapses can be enabled for STDP staying below this critical frequency. However, the number of STDP synapses should be decreased if a higher update controller frequency is required, e.g. for a configuration with an 8-bit weight resolution and a small number of SSPs.

Common resets of spike pair accumulations reduce synapse chip resources by requiring one instead of two reset lines, but suppress synaptic depression and bias the weight evolution towards potentiation. This is due to the feed-forward network architecture, in which causal relationships between pre- and postsynaptic spikes are more likely than anti-causal ones. Long periods of accumulation (large numbers of SSPs) lower the probability of synaptic depression. Hence, all weights tend to saturate at the maximum weight value impeding a distinction between both populations of synapses within the network benchmark (Figure 7G). The probability of synaptic depression can be increased by high weight update frequencies (small numbers of SSPs) shortening the accumulation periods (Equation 3) and subsequently approaching the behavior of independent resets. However, high weight update frequencies require high weight resolutions and thus high update controller frequencies, which decreases the number of available synapses enabled for STDP.

As a compensation for common resets, we suggest that the single spike pair accumulation threshold is expanded to multiple thresholds implemented as ADCs. In comparison to synapses with common resets, ADCs improve p-values significantly only for an 8-bit weight resolutions (Figure 7G, compare cyan to magenta values). However, the combination of two 4-bit hardware synapses allows to mimic independent resets and hence yields p-values comparable to 8-bit synapses using ADCs (Figure 7G, compare red to cyan values). Mimicking independent resets is under development for the FACETS wafer-scale hardware system. Each of the two combined synapses will be configured to accumulate only either causal or anti-causal spike pairs, while both synapses are updated in a common process. This requires only minor hardware design changes within the weight update controller and should be preferred to more expensive changes for realizing ADCs. The implementation of real second reset lines is not possible without major hardware design changes, but is considered for future chip revisions.

Benchmark simulations incorporating the measured variations within and between synapse circuits due to production imperfections result in p-values worse (higher) than for a 4-bit weight resolution (compare asterisk in Figure 8F to red value for $c = 0.025$ in Figure 7E). Consequently, a 4-bit weight resolution is sufficient for the current implementation of the measurement and accumulation circuits. We suppose that the isolatedly analyzed effects of production imperfections and weight discretization add up and limit the best possible p-value of each other. Analysis on combinations of hardware restrictions would allow to quantify how their effects add up and are considered for further studies. However, hardware variations can also be considered as a limitation on the transistor level making higher weight resolutions unnecessary.

Figure 9 summarizes the results on how to configure STDP on discrete weights. For a given weight resolution r the number n of SSPs has to be chosen as low as possible to allow for high weight update frequencies ν_w. However, n must be high enough to ensure STDP dynamics comparable to continuous weights (lightest gray shaded area) and to stay within the configuration space realizable by the FACETS wafer-scale hardware system. The hardware system limits the update controller frequency ν_c and hence distorts STDP especially for low n.

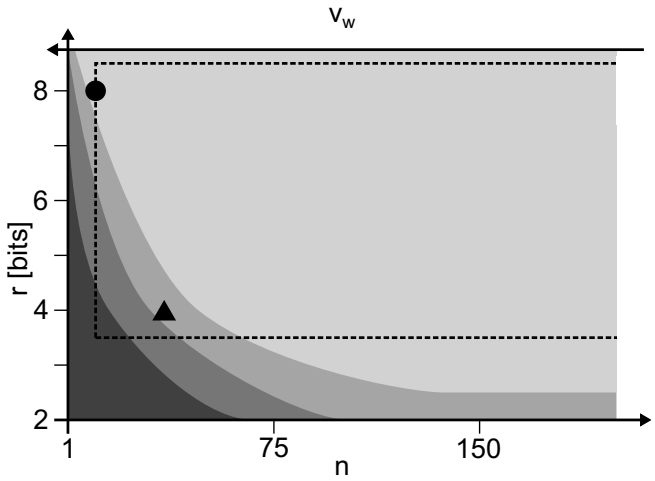


Figure 9: The configuration space of STDP on discrete weights spanned by the weight resolution r and the number n of SSPs that is inversely proportional to the weight update frequency ν_{w}. The darkest gray area depicts the configurations with dead discrete weights (Figure 3). The lower limits of configurations for proper equilibrium weight distributions (Figure 4) and single synapse dynamics (Figure 6) are shown with brighter shades. The dashed rectangle marks configurations realizable by the FACETS wafer-scale hardware system (assuming an acceleration factor of 10^3, all synapses enabled for STDP and SSPs applied with 10 Hz). The working points for a 4-bit ($n = 36$) and 8-bit ($n = 12$) weight resolution are highlighted as a triangle and circle, respectively.

4.4 Outlook

Currently, STDP in neuromorphic hardware systems is enabled for only 10 to few 10,000 synapses in real-time (Arthur & Boahen, 2006; Zou et al., 2006; Daouzli et al., 2008; Ramakrishnan et al., 2011). Large-scale systems do not implement long-term plasticity (Merolla & Boahen, 2006; Vogelstein et al., 2007) or operate in real-time only (Jin et al., 2010a). Enabling a large-scale (over $4 \cdot 10^7$ synapses) and highly accelerated neuromorphic hardware system (the FACETS wafer-scale hardware system) with configurable STDP requires trade-offs between number and size of synapses, which raises constraints in their implementation (Schemmel et al., 2006, 2010). Table 5 summarizes these trade-offs and gives an impression about the hardware costs and effects on STDP.

In this study, we introduced novel analysis tools allowing the investigation of hardware constraints and therefore verifying and improving the hardware design without the need for expensive and time-consuming prototyping. Ideally, this validation process should be shifted to an earlier stage of hardware design combining the expertise from Computational Neuroscience and Neuromorphic Engineering, as e.g. published by Linares-Barranco et al. (2011). This kind of research is crucial for researchers to use and understand research executed on neuromorphic hardware systems and thereby transform it into a tool substituting von Neumann computers in Computational Neuroscience. Brüderle et al. (2011) report the development of a *virtual hardware*, a simulation tool replicating the functionality and configuration space of the entire FACETS wafer-scale hardware system. This tool will allow further analyses on hardware constraints, e.g. in the communication infrastructure and configuration space.

The presented results verify the current implementation of the FACETS wafer-scale hardware system in terms of balance between weight resolution, update controller frequency and circuit variations. Further improvement of the existing hardware implementa-

Modification	Resource reduction	Effect on STDP
Global weight update controller	+++	Latency between synapse processings; spike pair accumulations necessary
Analog measurement of spike-timing-dependence	++	Analog measurements are affected by production imperfections
Reduced spike pairing scheme	++	n.a.
Decreased weight resolution	++	Loss in synapse dynamics and competition; large weight steps require spike pair accumulations
Reduction of operation frequency ν_c of the weight update controller (overall frequency could be increased by implementing multiple controllers)	++	Threshold over-shootings distorts synchrony detection
Common reset line	+	No synchrony detection possible
LUTs (compared to arithmetic operations)	+	None
ADCs as compensation for common resets	-	No significant compensation in case of 4-bit synapses

Table 5: Possible design modifications of hardware synapses, their reduction in terms of required chip resources and their effects on STDP. These modifications are listed by their resource reduction in descending order inspired by the FACETS wafer-scale hardware system and its production process. A larger reduction of chip resources allows more synapses on a single chip.

tion would require improvements of all aspects. The only substantial bottleneck has been identified to be common resets, already leading to design improvements of the wafer-scale system.

Although all presented studies refer to the intermediate Gütig STDP model, any other STDP model relying on Equation 1 and an exponentially decaying time-dependence can be investigated with the existing software tools in a generic way, e.g. those models listed in Table 1. In contrast to the fixed exponential time-dependence implemented as analog circuits in the FACETS wafer-scale hardware system, the weight-dependence is freely programmable and stored in a LUT.

Ideally, a high resolution in the weight range of highest plausibility is requested, a high *effective resolution*. Bounded STDP models (e.g. the intermediate Gütig STDP model applied in this study) are well suited for a 4-bit weight resolution and allow a linear mapping of continuous to discrete weights. A 4-bit weight resolution causes large weight updates and hence broadens the weight distribution spanning the whole weight range. This results in a high effective resolution. On the other hand, unbounded STDP models (e.g. the power law and van Rossum STDP models) have long tails towards high weights. Cutting the tail by only mapping low weights to discrete weights would increase the frequency of the highest discrete weight. A possible solution is a non-linear mapping of continuous to discrete weights - large differences between high discrete weights and small differences between low discrete weights. However, a variable distance between discrete weights would require more hardware efforts.

An all-to-all spike pairing scheme applied to the reference synapses within the network benchmark results in p-values worse (higher) than for synapses implementing a reduced symmetric nearest-neighbor spike pairing scheme (not shown, but comparable to 4-bit discrete weights in Figure 7E, see red values). Detailed analyses on different spike pairing schemes could be investigated in further studies.

As a next step, our hardware synapse model can replace the regular STDP synapses in simulations of established neural networks, to test their robustness and applicability for physical emulation in the FACETS wafer-scale hardware system. The synapse model is available in the following NEST release and can easily be applied to NEST or PyNN network descriptions. If neural networks, or modifications of them, qualitatively reproduce the simulation, they can be applied to the hardware system, with which similar results can be expected. Thus, the presented simulation tools allow beforehand modifications of network architectures to ensure the compatibility with the hardware system.

With respect to more complex long-term plasticity models, the hardware system is currently being extended by a programmable microprocessor that is in control of all weight modifications. This processor allows to combine synapse rows in order to compensate for common resets. With possible access to further neuron or network properties the processor would allow for more complex plasticity rules as e.g. those of Clopath et al. (2008) and Vogels et al. (2011). Even modifications of multiple neurons are feasible, a phenomenon observed in experiments with neuromodulators (Eckhorn et al., 1990; Itti & Koch, 2001; Reynolds & Wickens, 2002; Shmuel et al., 2005). Nevertheless, more experimental data and consensus about neuromodulator models and their applications are required to further customize the processor. New hardware revisions are rather expensive and consequently should only cover established models that are prepared for hardware implementation by dedicated studies.

This presented evaluation of the FACETS wafer-scale hardware system is meant to encourage neuroscientists to benefit from neuromorphic hardware without leaving their environment in terms of neuron, synapse and network models. We further endorse that,

towards an efficient exploitation of hardware resources, the design of synapse models will be influenced by hardware implementations rather than only by their mathematical treatability (e.g. Badoni et al., 2006).

5 Acknowledgment

The research leading to these results has received funding by the European Union 6th and 7th Framework Programme under grant agreement no. 15879 (FACETS) and no. 269921 (BrainScaleS). Special thanks to Yves Frégnac, Daniel Brüderle and Andreas Grübl for helpful discussions and technical support.

References

Amit, D., & Fusi, S. (1994). Learning in neural networks with material synapses. *Neural Comput. 6*(5), 957–982.

Arthur, J. V., & Boahen, K. (2006). Learning in silicon: Timing is everything. In *Advances in Neural Information Processing Systems (NIPS)*, Volume 18, Vancouver, pp. 75–82. MIT Press.

Badoni, D., Giulioni, M., Dante, V., & Del Giudice, P. (2006). An aVLSI recurrent network of spiking neurons with reconfigurable and plastic synapses. In *Proceedings of the 2006 International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, pp. 4. IEEE Press.

Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci. 18*, 10464–10472.

Bi, G., & Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annu. Rev. Neurosci. 24*, 139–66.

Bill, J., Schuch, K., Brüderle, D., Schemmel, J., Maass, W., & Meier, K. (2010). Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity. *Front. Comput. Neurosci. 4*(129).

Brader, J. M., Senn, W., & Fusi, S. (2007). Learning real world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput. 19*(11), 2881–2912.

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., Diesmann, M., Morrison, A., Goodman, P. H., Harris Jr., F. C., Zirpe, M., Natschläger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A. P., El Boustani, S., & Destexhe, A. (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *J. Comput. Neurosci. 23*(3), 349–398.

Brüderle, D., Petrovici, M., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., Grübl, A., Wendt, K., Müller, E., Schwartz, M.-O., Husmann de Oliveira, D., Jeltsch, S., Fieres, J., Schilling, M., Müller, P., Breitwieser, O., Petkov, V., Muller, L., Davison, A. P., Krishnamurthy, P., Kremkow, J., Lundqvist, M., Muller, E., Partzsch, J., Scholze, S., Zühl, L., Destexhe, A., Diesmann, M., Potjans, T. C., Lansner, A., Schüffny, R., Schemmel, J., & Meier, K. (2011). A comprehensive workflow for general-purpose neural

modeling with highly configurable neuromorphic hardware systems. *Biol. Cybern. 104*, 263–296.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci. 8*(3), 183–208.

Butts, D. A., Weng, C., Jin, J., Yeh, C.-I., Lesica, N. A., Alonso, J.-M., & Stanley, G. B. (2007). Temporal precision in the neural code and the timescales of natural vision. *Nature 449*(7158), 92–95.

Cassenaer, S., & Laurent, G. (2007). Hebbian STDP in mushroom bodies facilitates the synchronous flow of olfactory information in locusts. *Nature 448*(7154), 709–713.

Cassenaer, S., & Laurent, G. (2012). Conditional modulation of spike-timing-dependent plasticity for olfactory learning. *Nature 482*(7383), 47–52.

Clopath, C., Ziegler, L., Vasilaki, E., Büsing, L., & Gerstner, W. (2008). Tag-Trigger-Consolidation: A model of early and late long-term-potentiation and depression. *PLoS Comput. Biol. 4*(12), e1000248.

Daouzli, A., Saighi, S., Buhry, L., Bornat, Y., & Renaud, S. (2008). Weights convergence and spikes correlation in an adaptive neural network implemented on VLSI. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing*, Funchal, pp. 286–291.

Davison, A., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., & Yger, P. (2009). PyNN: a common interface for neuronal network simulators. *Front. Neuroinformatics 2*(11).

Davison, A. P., & Frégnac, Y. (2006). Learning cross-modal spatial transformations through spike timing-dependent plasticity. *J. Neurosci. 26*(21), 5604–5615.

Desbordes, G., Jin, J., Alonso, J.-M., & Stanley, G. B. (2010). Modulation of temporal precision in thalamic population responses to natural visual stimuli. *Front. Syst. Neurosci. 4*(151).

Desbordes, G., Jin, J., Weng, C., Lesica, N. A., Stanley, G. B., & Alonso, J.-M. (2008). Timing precision in population coding of natural scenes in the early visual system. *PLoS Biol. 6*(12), e324.

Eckhorn, R., Reitböck, H.-J., Arndt, M., & Dicke, P. (1990). Feature linking via synchronization among distributed assemblies: Results from cat visual cortex and from simulations. *Neural Comput. 2*, 293–307.

El Boustani, S., Yger, P., Frégnac, Y., & Destexhe, A. (2012). Stable learning in stochastic network states. *J. Neurosci. 32*(1), 194–214.

Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., & Burger, D. (2011). Dark silicon and the end of multicore scaling. In *Proceedings of the 2011 International Symposium on Computer Architecture (ISCA)*, San Jose, pp. 365–376. ACM Press.

FACETS (2010). Fast Analog Computing with Emergent Transient States, project website. Available at: `http://www.facets-project.org`.

Frégnac, Y. (2012). Personal communication.

Fromherz, P. (2002). Electrical interfacing of nerve cells and semiconductor chips. *ChemPhysChem 3*(3), 276–284.

Fusi, S., Drew, P. J., & Abbott, L. F. (2005). Cascade models of synaptically stored memories. *Neuron 45*(4), 599–611.

Gardiner, C. (2009). *Stochastic Methods: A Handbook for the Natural and Social Sciences* (4th ed.). Berlin, Heidelberg: Springer.

Gerstner, W., Kempter, R., van Hemmen, J. L., & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature 383*, 76–78.

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia 2*(4), 1430.

Gütig, R., Aharonov, R., Rotter, S., & Sompolinsky, H. (2003). Learning input correlations through nonlinear temporally asymmetric Hebbian plasticity. *J. Neurosci. 23*(9), 3697–3714.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Haefliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., & Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci. 5*(73). doi: 10.3389/fnins.2011.00073.

Indiveri, G., Stefanini, F., & Chicca, E. (2010). Spike-based learning with a generalized integrate and fire silicon neuron. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, pp. 1951–1954. IEEE Press.

Itti, L., & Koch, C. (2001). Computational modeling of visual attention. *Nat. Rev. Neurosci. 2*(3), 194–203.

Jin, X., Lujan, M., Plana, L., Davies, S., Temple, S., & Furber, S. (2010a). Modeling spiking neural networks on SpiNNaker. *Computing in Science Engineering 12*(5), 91–97.

Jin, X., Rast, A., Galluppi, F., Davies, S., & Furber, S. (2010b). Implementing spike-timing-dependent plasticity on SpiNNaker neuromorphic hardware. In *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)*, Barcelona, pp. 1–8. IEEE Press.

Johansson, C., & Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Networks 20*, 48–61.

Kempter, R., Gerstner, W., & van Hemmen, J. L. (1999). Hebbian learning and spiking neurons. *Phys. Rev. E 59*, 4498–4514.

Kuba, H., Koyano, K., & Ohmori, H. (2002). Synaptic depression improves coincidence detection in the nucleus laminaris in brainstem slices of the chick embryo. *Eur. J. Neurosci. 15*(6), 984–990.

Kuhn, A., Aertsen, A., & Rotter, S. (2003). Higher-order statistics of input ensembles and the response of simple model neurons. *Neural Comput. 1*(15), 67–101.

Kunkel, S., Diesmann, M., & Morrison, A. (2011). Limits to the development of feed-forward structures in large recurrent neuronal networks. *Front. Comput. Neurosci. 4*.

Levi, T., Lewis, N., Saighi, S., Tomas, J., Bornat, Y., & Renaud, S. (2008). *VLSI circuits for biomedical applications*, Chapter 12, pp. 241–264. Norwood: Artech House.

Levy, W. B., & Steward, D. (1983). Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus. *Neuroscience 8*, 791–797.

Linares-Barranco, B., Serrano-Gotarredona, T., Camunas-Mesa, L. A., Perez-Carrasco, J. A., Zamarreno-Ramos, C., & Masquelier, T. (2011). On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex. *Front. Neurosci. 5*.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Ann. Stat. 18*(1), 50–60.

Markram, H. (2006). The blue brain project. *Nat. Rev. Neurosci. 7*, 153–160.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science 275*, 213–215.

Marre, O., Yger, P., Davison, A. P., & Frégnac, Y. (2009). Reliable recall of spontaneous activity patterns in cortical networks. *J. Neurosci. 29*(46), 14596–14606.

Merolla, P., & Boahen, K. (2006). Dynamic computation in a recurrent network of heterogeneous silicon neurons. In *Proceedings of the 2006 International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, pp. 4539–4542. IEEE Press.

Millner, S., Grübl, A., Schemmel, J., Meier, K., & Schwartz, M.-O. (2010). A VLSI implementation of the adaptive exponential integrate-and-fire neuron model. In *Advances in Neural Information Processing Systems (NIPS)*, Volume 23, Vancouver, pp. 1642–1650.

Morrison, A., Aertsen, A., & Diesmann, M. (2007). Spike-timing dependent plasticity in balanced random networks. *Neural Comput. 19*, 1437–1467.

Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike-timing. *Biol. Cybern. 98*, 459–478.

Morrison, A., Mehring, C., Geisel, T., Aertsen, A., & Diesmann, M. (2005). Advancing the boundaries of high connectivity network simulation with distributed computing. *Neural Comput. 17*(8), 1776–1801.

Mu, Y., & Poo, M. (2006). Spike timing-dependent LTP/LTD mediates visual experience-dependent plasticity in a developing retinotectal system. *Neuron 50*(1), 115–125.

Nordlie, E., Gewaltig, M.-O., & Plesser, H. E. (2009). Towards reproducible descriptions of neuronal network models. *PLoS Comput. Biol. 5*(8), e1000456.

Pelgrom, M., Duinmaijer, A., & Welbers, A. (1989). Matching properties of MOS transistors. *IEEE J. Solid-St. Circ. 24*(5), 1433–1439.

Perrin, D. (2011). Complexity and high-end computing in biology and medicine. *Adv. Exp. Med. Biol. 696*, 377–384.

Plana, L. A., Furber, S. B., Temple, S., Khan, M., Shi, Y., Wu, J., & Yang, S. (2007). A GALS infrastructure for a massively parallel multiprocessor. *IEEE Des. Test Comput. 24*(5), 454–463.

Ramakrishnan, S., Hasler, P., & Gordon, C. (2011). Floating gate synapses with spike-time-dependent plasticity. *IEEE Trans. Biomed. Circuits Syst. 5*(3), 244 –252.

Reynolds, J. N., & Wickens, J. R. (2002). Dopamine-dependent plasticity of corticostriatal synapses. *Neural Networks 15*, 507–521.

Rubin, J., Lee, D., & Sompolinsky, H. (2001). Equilibrium properties of temporally asymmetric Hebbian plasticity. *Phys. Rev. Lett. 86*, 364–367.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, pp. 1947–1950. IEEE Press.

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

Schemmel, J., Fieres, J., & Meier, K. (2008). Wafer-scale integration of analog neural networks. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, Hong Kong. IEEE Press.

Schemmel, J., Gruebl, A., Meier, K., & Mueller, E. (2006). Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, pp. 1–6. IEEE Press.

Senn, W., Segev, I., & Tsodyks, M. (1998). Reading neuronal synchrony with depressing synapses. *Neural Comput. 10*(4), 815–819.

Shmuel, A., Korman, M., Sterkin, A., Harel, M., Ullman, S., Malach, R., & Grinvald, A. (2005). Retinotopic axis specificity and selective clustering of feedback projections from V2 to V1 in the owl monkey. *J. Neurosci. 25*(8), 2117–2131.

Sjöström, P., Turrigiano, G., & Nelson, S. (2001). Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron 32*, 1149–1164.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci. 3*(9), 919–926.

Thompson, S. E., & Parthasarathy, S. (2006). Moore's law: the future of si microelectronics. *Materials Today 9*(6), 20–25.

Trachtenberg, J. T., Chen, B. E., Knott, G. W., Feng, G., Sanes, J. R., Welker, E., & Svoboda, K. (2002). Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature 420*, 788–794.

Tsodyks, M. V., & Markram, H. (1997). The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proc. Natl. Acad. Sci. USA 94*, 719–723.

Turrigiano, G. G., Leslie, K. R., Desai, N. S., Rutherford, L. C., & Nelson, S. B. (1998). Activity-dependent scaling of quantal amplitude in neocortical neurons. *Nature 391*, 892–896.

van Rossum, M. C. W., Bi, G., & Turrigiano, G. G. (2000). Stable Hebbian learning from spike timing-dependent plasticity. *J. Neurosci. 20*(23), 8812–8821.

Vogels, T., Sprekeler, H., Zenke, F., Clopath, C., & Gerstner, W. (2011). Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science 334*(6062), 1569–1573.

Vogels, T. P., Rajan, K., & Abbott, L. F. (2005). Neural network dynamics. *Annu. Rev. Neurosci. 28*, 357–376.

Vogelstein, R., Tenore, F., Guevremont, L., Etienne-Cummings, R., & Mushahwar, V. (2008). A silicon central pattern generator controls locomotion in vivo. *IEEE Trans. Biomed. Circuits Syst. 2*(3), 212–222.

Vogelstein, R. J., Mallik, U., Culurciello, E., Cauwenberghs, G., & Etienne-Cummings, R. (2007). A multichip neuromorphic system for spike-based visual information processing. *Neural Comput. 19*(9), 2281–2300.

Yger, P., El Boustani, S., Destexhe, A., & Frégnac, Y. (2011). Topologically invariant macroscopic statistics in balanced networks of conductance-based integrate-and-fire neurons. *J. Comput. Neurosci. 31*, 229–245.

Zou, Q., Bornat, Y., Saighi, S., Tomas, J., Renaud, S., & Destexhe, A. (2006). Analog-digital simulations of full conductance-based networks of spiking neurons with spike timing dependent plasticity. *Network: Comput. Neural Systems 17*(3), 211–233.

6 Appendix

6.1 Analytical distributions

Weight evolutions can be described by asymmetric Markov processes with boundary conditions. Following van Rossum et al. (2000), the weight distribution $P(w)$ can be expressed by a Taylor expansion of the underlying master equation

$$\frac{\partial P(w,t)}{\partial t} = -p_\mathrm{d} P(w,t) - p_\mathrm{p} P(w,t) + p_\mathrm{d} P(w + \Delta w_\mathrm{d}, t) + p_\mathrm{p} P(w - \Delta w_\mathrm{p}, t). \tag{4}$$

In contrast to van Rossum et al. (2000), this study defines a weight step Δw by a sequence of n weight updates δw as described by Equation 1. Hence the weight steps Δw can be written as $\Delta w_\mathrm{d}(w) = (w + F_-(w))_n - w$ and $\Delta w_\mathrm{p}(w) = (w + F_+(w))_n - w$, where $f(w)_n$ is the n-th recursive evaluation of $f(w)$.

According to van Rossum et al. (2000) this Taylor expansion results in the Fokker-Planck equation

$$\frac{\partial P(w,t)}{\partial t} = -\frac{\partial}{\partial w}\left[A(w)P(w,t)\right] + \frac{1}{2}\frac{\partial^2}{\partial w^2}\left[B(w)P(w,t)\right] \tag{5}$$

with jump moments $A(w) = p_\mathrm{d}\Delta w_\mathrm{d}(w) + p_\mathrm{p}\Delta w_\mathrm{p}(w)$ and $B(w) = p_\mathrm{d}\Delta w_\mathrm{d}(w)^2 + p_\mathrm{p}\Delta w_\mathrm{p}(w)^2$, which has the following solution for reflecting boundary conditions (Gardiner, 2009):

$$P(w) = \frac{N}{B(w)} \exp\left[2\int_0^w \frac{A(w')}{B(w')} dw'\right], \tag{6}$$

with N as a normalization factor. For small n this equation can be solved analytically, but is integrated numerically to cover also large n.

However, this analytical approach fails, because the Taylor expansion in combination with the boundary conditions does not hold for large n (absorbing boundary conditions do not improve the results).

6.2 STDP in the FACETS chip-based hardware system

The STDP mechanism of the FACETS chip-based hardware system differs from that of the FACETS wafer-scale hardware system as follows. The major difference is the comparison of spike pair accumulations with thresholds. The wafer-scale system analyzed in this study compares both spike pair accumulations with a threshold (the threshold can be set independently for both accumulations, but they are assumed to be equal in this study). An weight update is performed if a single accumulation crosses this threshold. In contrast, the chip-based system used for all measurements subtracts both spike pair accumulations and compares the absolute value of their difference $|a_\mathrm{c} - a_\mathrm{a}|$ with a single threshold. If this threshold is crossed, the sign of the difference between the spike pair accumulations $\mathrm{sig}\,(a_\mathrm{c} - a_\mathrm{a})$ determines, whether the causal or anti-causal accumulation prevails and the weight is updated accordingly. However, this difference between both hardware systems can be neglected, because both STDP mechanisms are identical if exclusively causal or anti-causal spike pairs are accumulated. This is the case for the measurement protocol of STDP curves.

Parameter	Description	Value
V_{clrc}	Amount of charge that will be accumulated on the capacitor C_1 (Schemmel et al., 2006) in case of causal spike time correlations, corresponds to $x(\Delta t)$	0.90 V
V_{clra}	See V_{clrc}, but for the anti-causal circuit	0.94 V
V_{ctlow}	Lower spike pair accumulation threshold	0.85 V
V_{cthigh}	Higher spike pair accumulation threshold	1.0 V
$adjdel$	Adjustable delay between the pre- and postsynaptic spike	2.5 μA
V_{m}	Parameter to stretch the STDP time constant τ_{STDP}	0.0 V
$I_{\mathrm{bcorreadb}}$	Bias current that influences timing issues during read outs	2.0 μA
$drvI_{\mathrm{rise}}$	Rise time of synaptic conductance	1.0 V
$drvI_{\mathrm{fall}}$	Fall time of synaptic conductance	1.0 V
V_{start}	Start value of synaptic conductance, need for small rise times	0.25 V
$drvI_{\mathrm{out}}$	Maximum value of synaptic conductance, corresponds to g_{max}	variable

Table 6: Applied hardware parameters. The difference $V_{\mathrm{cthigh}} - V_{\mathrm{ctlow}}$ corresponds to the threshold a_{th}. All data is recorded with the FACETS chip-based hardware system using chip number 444 and synapse column 4.

6.3 Generating spike pairs in hardware

Spike pairs in the FACETS chip-based hardware system are generated as follows. Presynaptic spike times can be set precisely, whereas postsynaptic spikes need to be triggered by presynaptic input. Therefore, a presynaptic spike (via the measured synapse) and m trigger spikes (eliciting a postsynaptic spike) are fed into a single neuron occupying $m+1$ synapses. The synaptic weights as well as the synapse driver strengths of the trigger synapses are proportional to the synaptic peak conductance and are adjusted in such a way that a single postsynaptic spike is evoked. The highest reliability of spike times within a hardware run and between runs is achieved for $m = 4$ trigger synapses (not shown here). The synapse driver strength is set to the intermediate value between the limiting case of no and multiple postsynaptic spikes evoked by one trigger only. The synaptic weight of the measured synapse is set to zero and consequently the measured synapse has no influence on the elicitation of postsynaptic spikes.

A: Model summary	
Populations	three: uncorrelated input (U), correlated input (C), target (T)
Topology	feed-forward
Connectivity	all-to-one
Neuron model	leaky integrate-and-fire, fixed voltage threshold, fixed absolute refractory period (voltage clamp)
Synapse model	exponential-shaped postsynaptic conductances
Plasticity	intermediate Gütig spike-timing dependent plasticity
Input	fixed-rate Poisson (for U) and multiple interaction process (for C) spike trains
Measurements	synaptic weights

B: Populations		
Name	**Elements**	**Population size**
U	parrot neurons	N_{U}
C	parrot neurons	N_{C}
T	iaf neurons	N_{T}

C: Connectivity		
Source	**Target**	**Pattern**
U	T	all-to-all, uniformly distributed initial weights w, STDP, delay d
C	T	

D: Neuron and synapse model	
Name	iaf neuron
Type	leaky integrate-and-fire, exponential-shaped synaptic conductances
Sub-threshold dynamics	$C_{\mathrm{m}}\frac{\mathrm{d}V}{\mathrm{d}t} = g_{\mathrm{L}}(E_{\mathrm{L}} - V) + g(t)(E_{\mathrm{e}} - V)$ if $t > t^* + \tau_{\mathrm{ref}}$ $V(t) = V_{\mathrm{reset}}$ else $g(t) = w g_{\max} \exp(-t/\tau_{\mathrm{syn}})$
Spiking	If $V(t-) < \theta \wedge V(t+) \geq \theta$ 1. set $t^* = t$, 2. emit spike with time stamp t^*
Name	parrot neuron
Type	repeats input spikes with delay d

E: Plasticity	
Name	intermediate Gütig STDP
Spike pairing scheme	reduced symmetric nearest-neighbor
Weight dynamics	$\delta w(w, \Delta t) = F(w)x(\Delta t)$ $x(\Delta t) = \exp(-\lvert\Delta t\rvert/\tau_{\mathrm{STDP}})$ $F(w) = \lambda(1 - w)^{\mu}$ if $\Delta t > 0$ $F(w) = -\lambda\alpha w^{\mu}$ if $\Delta t < 0$

F: Input		
Type	**Target**	**Description**
Poisson generators	U	independent Poisson spike trains with firing rate ρ
MIP generators	C	spike trains with correlation c and firing rate ρ

G: Measurements
evolution and final distribution of all synaptic weights

Table 7: Model description of the network benchmark using the reference synapse model. After Nordlie et al. (2009). For details about the hardware-inspired synapse model see Section 2.6.1.

B: Populations		
Name	**Value**	**Description**
N_{U}	10	number of neurons in uncorrelated input population
N_{C}	10	number of neurons in correlated input population
N_{T}	1	number of neurons in target population
C: Connectivity		
Name	**Value**	**Description**
w	uniformly distributed over [0,1]	number of neurons in uncorrelated input population
d	0.1 ms	synaptic transmission delays
D: Neuron and synapse model		
Name	**Value**	**Description**
C_{m}	250 pF	membrane capacity
g_{L}	16.6667 nS	leakage conductance
E_{L}	−70 mV	leakage reversal potential
θ	−55 mV	fixed firing threshold
V_{reset}	−60 mV	reset potential
τ_{ref}	2 ms	absolute refractory period
E_{e}	0 mV	excitatory reversal potential
g_{max}	100 nS	postsynaptic maximum conductance
τ_{syn}	0.2 ms	postsynaptic conductance time constant
E: Plasticity		
Name	**Value**	**Description**
α	1.05	asymmetry
λ	0.005	learning rate
μ	0.4	exponent
τ_{STDP}	20 ms	STDP time constant
F: Input		
Name	**Value**	**Description**
ρ	7.2 Hz	firing rate
c	[0.005,0.05]	pair-wise correlation between spike trains

Table 8: Parameter specification. The categories refer to the model description in Table 7.

III Neuromorphic learning towards nano second precision

Thomas Pfeil[1,†], Anne-Christine Scherzer[1], Johannes Schemmel[1], and Karlheinz Meier[1]

[1]*Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany*

Abstract

Temporal coding is one approach to representing information in spiking neural networks. An example of its application is the location of sounds by barn owls that requires especially precise temporal coding. Dependent upon the azimuthal angle, the arrival times of sound signals are shifted between both ears. In order to determine these interaural time differences, the phase difference of the signals is measured. We implemented this biologically inspired network on a neuromorphic hardware system and demonstrate spike-timing dependent plasticity on an analog, highly accelerated hardware substrate. Our neuromorphic implementation enables the resolution of time differences of less than 50 ns. On-chip Hebbian learning mechanisms select inputs from a pool of neurons which code for the same sound frequency. Hence, noise caused by different synaptic delays across these inputs is reduced. Furthermore, learning compensates for variations on neuronal and synaptic parameters caused by device mismatch intrinsic to the neuromorphic substrate.

[†]Received funding by the European Union 7th Framework Programme under grant agreement no. 243914 (Brain-i-Nets).

1 Introduction

Phase-locking has been shown to be one approach towards precise temporal coding in neural information processing [4] and is observed in the auditory pathway of barn owls [3]. Barn owls locate sounds by measuring the difference between the respective arrival times at both ears, the so-called interaural time difference (ITD), also known as the Jeffress model [10]. A neuron in the laminar nucleus will fire at a high rate if it detects coincidences between the two periodic signals that code the same sound frequency at both ears (Figure 1B and C). In other words, the neuron will fire at a maximum rate if the signals from both ears arrive coherently. This requires spike times that are "locked" to a specific phase of the sound signal. The more precisely spikes are locked, the higher the temporal resolution for measuring ITDs is. Previous studies have shown phase-locking with precision much smaller than the membrane time constants of the involved neurons [3]. However, synaptic delays differ across the neurons that code for the same sound frequency [3]. Coherence in the arrival time of signals can be restored by learning transmission delays [20] or by selecting synapses with simultaneous activity [4].

In this study, we present an analog, neuromorphic implementation of a spiking neural network which selects inputs out of a broad distribution of transmission delays by means of an unsupervised, on-chip Hebbian learning rule. The intrinsic, high acceleration factor of the neuromorphic substrate [16] allows for learning of phase-locking with $100\,\mathrm{ns}$ precision in hardware time domain. Finally, the results of this on-chip synapse selection is applied to detect ITDs of less than $50\,\mathrm{ns}$.

In addition to noise induced by variations in transmission delays, device mismatch in analog circuitries of neurons and synapses causes fixed-pattern noise on neural components. This means, parameters vary between neurons and synapses, as, for example, in the membrane time constant and synaptic strength.

Both types of variations can be reduced by off-chip calibration routines. In this study, they are compensated by on-chip learning mechanisms. In contrast, an implementation without plasticity, but the same variations in neural components, can barely measure any time differences between two $1\,\mathrm{MHz}$ signals in hardware time domain.

One of the first neuromorphic implementations for coincidence detection was a silicon replication of the auditory pathway [12]. Sound location is implemented by two synapses which transmit the signal from each ear. However, this device does not support on-chip learning, and connectivity between neurons is hard-wired. In this study, each ear is represented by a population of synapses rather than a single synapse. Inspired by the biological example [3] we add noise to the synaptic delays of this population. We demonstrate that spike-timing dependent plasticity which is implemented in our neuromorphic hardware system de-noises the input. The novelty of this system is to combine a dense integration of highly accelerated neurons with on-chip learning in each synapse [16] that allows for learning of coincidence detection with resolution higher than $50\,\mathrm{ns}$. Other approaches with re-configurable connectivity and on-chip plastic synapses have lower counts of neurons and synapses [8, 1, 6], mostly optimizied for low power consumption, or operate in biological real-time [21].

2 Network and hardware description

The neuromorphic hardware system we used is designed as a re-configurable, universal neuromorphic substrate on which neural networks operate 10^4 times faster than biological real-time (Figure 1A, [16]). It comprises pair-wise spike-timing dependent plasticity (STDP) in each of up to 256 synapses per neuron [19]. This type of Hebbian learning rule is adapted from measurements in biological tissue [14, 2, 15] and is described later in this section.

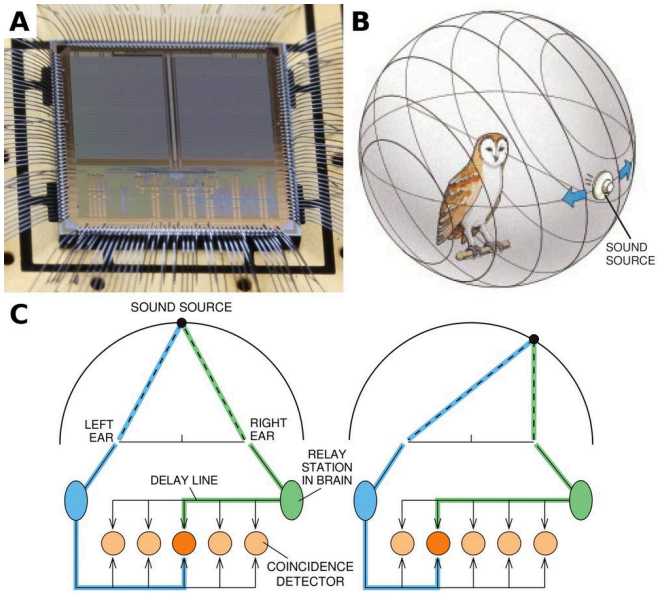


Figure 1: **(A)** Microphotograph of the neuromorphic chip (fabricated in a 180 nm CMOS process with die size $5 \times 5\,\mathrm{mm}^2$). **(B)** A barn owl locates sound by measuring the interaural time difference of a presented sound signal. **(C)** Schematic of the auditory pathway of barn owls. Neurons in the laminar nucleus detect coincidences between signals that arrive from both ears through fibres serving as delay lines (blue and green lines). The firing rate of the neuron at which the signals arrive simultaneously has the highest firing rate (colored darkly). Consequently, each neuron codes for an azimuthal sector. Figure (B) and (C) are adapted from [11].

The network model implemented *in silico* is inspired by the auditory pathway of barn owls [3, 4]. In this study, we investigate sound processing of a single frequency channel, exemplarily for any frequency of comparable scale. A neuron in the nucleus laminaris receives bilateral input from several neurons in the cochlear nucleus magnocellularis (Figure 2). The network selects those inputs, of which signals arrive coherently at the postsynaptic neuron, in order to improve the temporal coding of sound signals which is used for sound localization.

The delay of signal transmission Δd_i for each connection with index i is Gaussian distributed with standard deviation σ [3]. Note that $\sigma \cdot 2$ is larger than the period T of the input signal and, consequently, neighboring volleys of spikes overlap (black and blue spikes in Figure 2). The postsynaptic neuron is mimicked by a hardware neuron that approximates the conductance-based, leaky integrate-and-fire neuron model [16, 9]. The presynaptic input is modeled by individual spike trains fed into the system. Figure 2 shows the allocation of hardware resources including the synaptic nodes, each of which comprises STDP.

As the membrane time constant of hardware neurons can not be configured to values below $\tau_\mathrm{m}^\mathrm{hw} \approx 200\,\mathrm{ns}$ in hardware time domain, we assume an acceleration factor of 500 throughout this study for better comparison with biological measurements. Thus, all time constants, as well as experiment time, are given in biological time domain, if not stated otherwise. For example, in the following, we use a neuron with $\tau_\mathrm{m}^\mathrm{hw} = 200\,\mathrm{ns}$ in hardware time domain which translates to $\tau_\mathrm{m} = 100\,\mu\mathrm{s}$ in biological time domain, as suggested in [4].

We stimulate the network with a pure tone similar to [4]. For each presynaptic input spikes are drawn with probability $p_\mathrm{spike} = 0.35$ from a template of regular spike times with frequency $f = 2\,\mathrm{kHz}$ in biological time domain. Additionally, each spike has a jitter following a Gaussian distribution with standard deviation $40\,\mu\mathrm{s}$. The individual transmission delay Δd_i for each input with index i is modeled by being added to all spike times of this input.

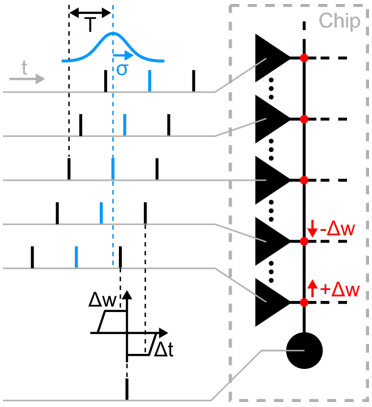


Figure 2: Network implementation on the neuromorphic chip. The postsynaptic hardware neuron (black circle) receives input from 64 presynaptic spike sources, whose spike times are generated on the host computer and transferred to the chip. All inputs (black triangles) show regular spike times of identical frequency $f = \frac{1}{T} = 2\,\text{kHz}$, but each input is shifted in time following a Gaussian distribution with $\sigma = 300\,\mu\text{s}$ (blue). For simplicity we neglected $p_{\text{spike}} < 1$ and jitter on spike times in this schematic. When the postsynaptic neuron fires, each synapse (red circles) measures the correlations (Δt) between its pre- and postsynaptic spikes. During the next evaluation of these synapses, their weights w are potentiated or depressed according to an additive rule ($\Delta w = \pm 1$, respectively).

On hardware, the strength of a synaptic connection g_i is the product of a conductance g_i^{max} adjustable for each input stream (triangles in Figure 2) and a 4-bit digital weight w_i stored in each synapse:

$$g_i = g_i^{max} w_i \tag{1}$$

While g_i^{max} is static throughout an emulation run, w_i is subject to STDP and can assume integer values between 0 and 15. However, g_i^{max} varies between synapses. This is caused by device mismatch due to imperfections in the production process [16]. Excitatory postsynaptic potentials (EPSPs) were recorded by measuring the impact of a single spike on the resting state of the postsynaptic membrane potential V_{m} (Figure 3A). Their time course is configured to be as short as possible, and their amplitude (g^{max}) is set to an intermediate value. This results in a target firing rate of approximately 1 kHz in the final network implementation.

Synaptic weights w_i are modified by on-chip learning mechanisms at accelerated runtime described as follows: The temporal correlations between spike pairs are measured and stored locally in each synapse by analog circuitry. Correlations between pre-post and post-pre spike pairs are accumulated as charge on two capacitors, respectively. In contrast to the local measurement and accumulation, the evaluation of these measurements is performed by controllers shared between synapses. Thereby, the controller compares for each synapse the amount of charge on its capacitors as follows:

$$|a_{\text{c}} - a_{\text{a}}| > a_{\text{th}} \tag{2}$$

$$a_{\text{c}} - a_{\text{a}} > 0 \tag{3}$$

where a_{c} is the charge on the capacitor for pre-post, a_{a} for post-pre spike pairs, and a_{th} a configurable threshold. If Equation 2 is true the synaptic weight is updated and the capacitors are discharged. Otherwise the weight stays unchanged and correlations are further accumulated without discharge of the capacitors. If a weight update is elicited, the synaptic weight will be increased by one if Equation 3 is true, otherwise it is decreased by one (in both cases with absorbing boundaries at the minimum and maximum value). In fact, hardware STDP is not limited to this additive

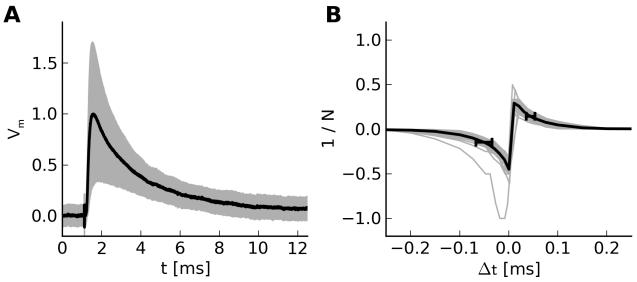


Figure 3: EPSPs and STDP learning windows of 64 hardware synapses. **(A)** Each EPSPs is averaged over 100 runs. The mean and standard deviation over all synapses are depicted in black and gray, respectively. The membrane potential V_m is plotted in arbitrary units. The area under these EPSPs has a ratio of standard deviation to mean of $\approx 50\%$. **(B)** Recording STDP learning windows (gray) is summarized as follows: The inverse number $1/N$ of spike pairs that need to be accumulated until a weight update is elicited is plotted against the time difference Δt between the pre- and postsynaptic spike (details are described elsewhere [17]). A value of $N = 1$ means that one pre-post pair is already sufficient to mark this synapse to be updated. The mean over all synapses and errors at half maximum (0.044 ± 0.009ms and -0.050 ± 0.016ms) are depicted in black.

rule, but can be configured to any rule via look-up tables. In the network presented, processing one synapse takes $1.5\,\mu$s in hardware time domain, which has been shown to be sufficient for coincidence detection in small networks [17]. For a detailed description of the implementation and configuration of hardware STDP see [19] and [17].

Parameters for the learning window of STDP on hardware must be adjusted to meet two criteria: On the one hand, the window should be broad in order to resolve a full period T of the input signal and to ensure a sufficiently high learning rate. On the other hand, the number N of spike pairs for close-by Δt should be distinguishable which is the case for overall large N. In Figure 3B a trade-off between both criteria is shown, because they can not be fulfilled together on the hardware. Note that learning windows are subject to device mismatch too, due to their analog measurement and accumulation circuitry. The width at half maximum is approximately $0.05\,$ms in biological time domain. The time between successive weight updates for 64 inputs is $48\,$ms.

The experiment protocol is split into two steps. First, the network is stimulated without any interaural time difference and on-chip plasticity is activated. The network selects appropriate inputs by altering on-chip, synaptic weights. Second, these learned synaptic weights are adopted for subsequent emulations, during which plasticity is turned off. The ITD is varied and sound is located via the firing rate of the postsynaptic neuron.

In the first step, the spiking neural network shown in Figure 2 is emulated for a duration of $10\,$s, while spike times of the postsynaptic neuron are recorded. After emulation, the digital weights of all synapses are read out from the chip.

Network performance is measured by the vector strength ν, which quantifies the precision of phase-locking at the postsynaptic site [5]. Each spike time t_i can be considered as a vector of unit length with a phase angle $\Theta_i = 2\pi f t_i$ as well as x and y components as follows:

$$(x_i, y_i) = (\sin(\Theta_i), \cos(\Theta_i)) \tag{4}$$

The vector strength is defined as the length of the mean vector across the overall number N of postsynaptic spike times:

$$\nu = \frac{1}{N}\sqrt{\left(\sum\nolimits_{i=1}^{N} x_i\right)^2 + \left(\sum\nolimits_{i=1}^{N} y_i\right)^2} \tag{5}$$

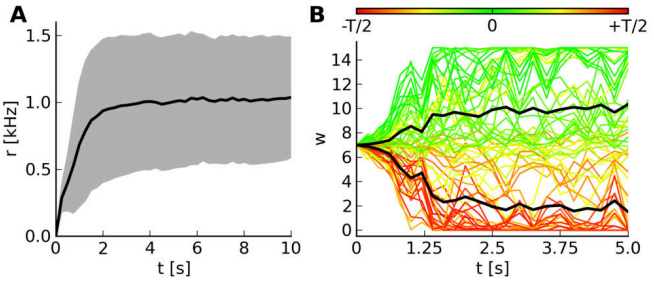


Figure 4: Learning process: Development of postsynaptic firing rates as well as synaptic weights over time. **(A)** The mean firing rate r and standard deviation of 100 emulations with 10 s duration in black and gray, repectively. Transmission delays Δd and input spike times are re-drawn for each emulation. **(B)** Development of digital hardware weights w over time for each synapse. For technical convenience, each set of weights at time t is recorded after an emulation with duration t. For each time step weights are averaged over 20 emulations, all with the same distribution of Δd and input spike times. The color code indicates the difference to the average phase of the postsynaptic neuron.

It can assume values between 0 and 1. It is minimal for randomly distributed spikes over time, and maximal for interspike intervals that are multiples of T. The angular dispersion of the mean vector translates to temporal precision [7]:

$$\sigma_{\mathrm{PL}} = \frac{\sqrt{2(1-\nu)}}{2\pi f} \tag{6}$$

In the second step, the sensitivity of the postsynaptic neuron's firing rate to time differences between two input signals, e.g. ITDs, is determined as follows: First, signal transmission delays and synaptic weights are adopted from the learning result of an emulation described above. Second, afferent connections are randomly divided into two groups of equal size. Third, time differences are modeled by adding an additional transmission delay to one of these groups. Finally, this network is emulated with static synaptic weights, and the firing rate of the neuron is recorded.

3 Hardware emulation results

Network emulations on hardware show precise, phase-locked spiking of the postsynaptic neuron. At the beginning of an emulation, all synapses have the same weight, and the firing rate of the postsynaptic neuron is low. The firing rate increases with increasing weights of those inputs that drive the neuron most (compare Figure 4A to green traces in B). Once strong synapses have evolved, phase-locking improves (ν increases, not shown) and synapses firing out of phase are weakend (red traces in Figure 4B). After approximately 3 s, the strong and weak synapses are in balance, and the postsynaptic firing rate saturates. The variance of firing rates between emulations (Figure 4A) is mostly caused by re-drawing the transmission delays for each emulation. Variations on synapses bias the learning process. For example, few strong synapses with improbable ($|\Delta d| \gg 0$ in upper plot of Figure 5A), but similar Δd may outperform many weak synapses with probable ($\Delta d \approx 0$), but similar Δd. Nevertheless, learning compensates for these variations.

Synapses will be termed *selected* if their weight after emulation exceeds their initial weight. Synapses with similar transmission delays or delays which differ by multiples of T are active simultaneously, and preferably drive the postsynaptic neuron. If a postsynaptic spike is elicited by these inputs, they are selected by the on-chip learning mechanism (compare Figure 5A to Figure 4B). This periodical selection scheme with period T is due to the overlap of consecutive spike

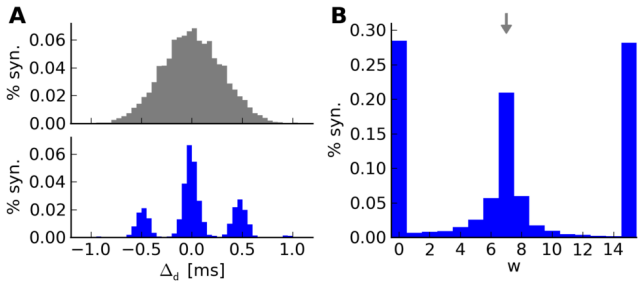


Figure 5: Selection of synapses and weights after on-chip learning. **(A)** Top: The distribution of signal transmission delays Δd before learning that is accumulated over the emulations shown in Figure 4A. The distribution is normalized such that its mean is 0. Bottom: The same distribution after learning, but only synapses with $w > w_{\text{start}}$ are shown. The phase obtained by summing up the phase vectors (Equation 4) of all spikes is subtracted from the transmission delays after each emulation. In other words, transmission delays are normalized such that the postsynaptic neuron fires preferably at $\Delta d = 0$. **(B)** The distribution of digital hardware weights w after learning for the same emulations as in (A). Before learning all weights were set to $w_{\text{start}} = 7$ (arrow).

volleys (Figure 2). However, not all synapses saturate to the minimum or maximum weight (Figure 5B). Some synaptic weights stay at their initial value. This can be explained by the technical implementation of STDP on hardware. If both correlations, pre-post and post-pre, accumulate at the same rate, no weight update will be triggered because the update controller evaluates the difference between both accumulations (see Equation 2 and [19]).

As an application, we show the detection of ITDs. To this end, we split the resulting synapses of a single run of Figure 5 into two groups, one for the input from each ear. Both, selected and unselected synapses are adopted. The performance of phase-locking with $\Delta_{\text{ITD}} = 0$ is shown in Figure 6A. Postsynaptic spikes occur preferably at the same phase with a vector strength of $\nu = 0.89$ which translates to a precision of $\sigma_{\text{PL}} \approx 40\,\mu\text{s}$ in biological time domain. Shifting the input of one group by Δ_{ITD} deteriorates the precision of phase-locking and consequently reduces the postsynaptic firing rate (Figure 6C). At $\Delta_{\text{ITD}} = \frac{T}{2}$ the postsynaptic neuron receives alternating input from both groups, and the vector strength is on the same level as the control (compare Figure 6C to D). For the control we applied the same protocol as before (see Figure 6A and C), but with a uniform weight distribution instead of previously learned synaptic weights. Time differences of less than 25 µs can be resolved by the firing rate of the postsynaptic neuron (compare error bars of neighboring data points of thin red line in Figure 6C). In contrast, the firing rate of the control barely has a dependency on Δ_{ITD} (thin red line in Figure 6D). This makes it difficult, if not impossible, to determine any time differences of 2 kHz signals.

4 Conclusions

We have presented an analog, neuromorphic network implementation that de-noises phase information and thereby learns to resolve time differences between two periodic stimuli of less than 50 ns in hardware time domain. De-noising is realized by unsupervised, on-chip spike-timing dependent plasticity that improves coincidence detection and locks spike times to a specific phase of the input signal. This results in precise phase information at the neuron site which enables the resolution of short phase differences, as for example those of sound signals from both ears. The network performance is comparable to similar network models simulated in software (compare Figure 6C to Figure 7). However, the absolute values for firing rate and vector strength differ due

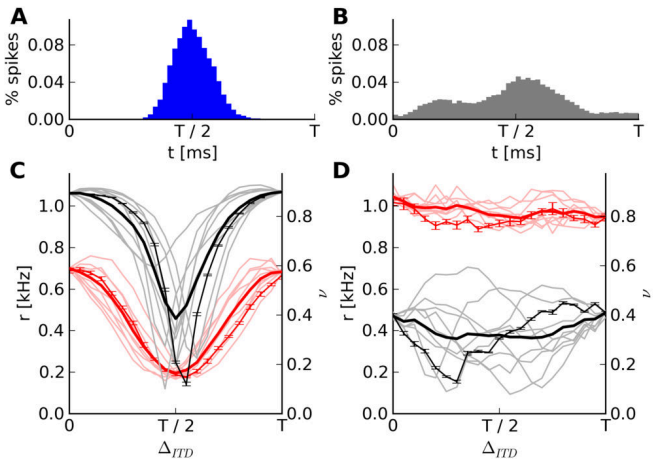


Figure 6: Phase-locking and the detection of interaural time differences. **(A)** Cyclic peristimulus time histograms (PSTHs) of postsynaptic spike times for one arbitrary emulation of those analyzed in Figure 5 ($T = 0.5\,\text{ms}$). The vector strength is $\nu = 0.89$ ($\approx 40\,\mu\text{s}$ precision), and the average vector strength over all 100 emulations is $\bar{\nu} = 0.81 \pm 0.10$ ($\approx 50\,\mu\text{s}$ precision). **(B)** PSTH for emulations of the same network and input as in (A), but STDP deactivated and w_{start} adjusted to obtain similar postsynaptic firing rates ($\nu = 0.38$ and $\bar{\nu} = 0.43 \pm 0.19$). **(C)** Firing rate r (bold red) and vector strength ν (bold black) averaged over 10 emulations for one half of synapses receiving input delayed by Δ_{ITD} compared to the other. Single emulations are shown in light shades, each with another random division of the synapses, but the same spike times. For one arbitrary random division the mean over 5 emulations is shown in thin black lines. Thereby, the standard deviation is the trial-to-trial variability. The network has the same transmission delays and synaptic weights as in (A). **(D)** The same protocol as in (C), but with static synaptic weights adopted from (B).

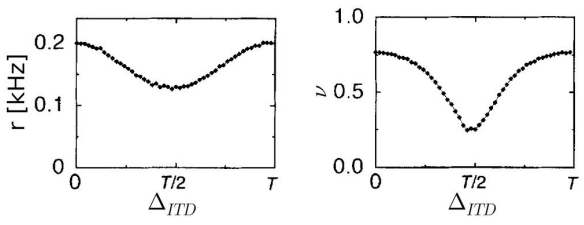


Figure 7: Results of a similar network as described in this study, but simulated in software (adopted from [4]). The firing rate r and vector strength ν of the network is shown in dependence on the interaural time difference Δ_{ITD}. The signal frequency is 5 kHz.

to different neuron, synapse, and STDP models, variations of model parameters on hardware, as well as a higher signal frequency in the reference publication [4].

Additionally, learning does not only de-noise the input, but also compensates for variations between neural components (Figure 3). These variations are caused by device mismatch and are inherent in all neuromorphic systems with analog circuitry. Intrinsically weak synapses with simultaneous impact on the membrane potential can outperform intrinsically strong synapses. This allows the measurement of short time differences, although the input is noisy and the neuromorphic substrate has variations in its neural components. Performance may even be improved by a preceding off-chip calibration of synaptic strengths. Furthermore, population coding reduces noise within signal transmissions by averaging across many unreliable components.

Although variations of neuronal components are measured in biology [13], it is still unclear how robust a neural network has to be in order to perform computation on these components. Large-scale neuromorphic systems, as described in [18], may particularly benefit from such self-adjusting, and hence robust, network implementations. In further studies, the neuromorphic network could be embedded into robotic systems for processing sensory data of, for example, ultrasonic sound. This would exploit the high acceleration factor of the neuromorphic system and its robust capability for handling noise and variations of neural components.

References

[1] Badoni, D., Giulioni, M., Dante, V., & Del Giudice, P. (2006). An aVLSI recurrent network of spiking neurons with reconfigurable and plastic synapses. In *Proceedings of the 2006 International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, pp. 4. IEEE Press.

[2] Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci. 18*, 10464–10472.

[3] Carr, C., & Konishi, M. (1990). A circuit for detection of interaural time differences in the brain stem of the barn owl. *J. Neurosci. 10*, 3227–3246.

[4] Gerstner, W., Kempter, R., van Hemmen, J. L., & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature 383*, 76–78.

[5] Goldberg, J. M., & Brown, P. B. (1969). Response of binaural neurons of dog superior olivary complex to dichotic tonal stimuli: some physiological mechanisms of sound localization. *J. Neurophysiol. 32*(4), 613–36.

[6] Hafliger, P. (2007). Adaptive WTA with an analog VLSI neuromorphic learning chip. *IEEE Trans. Neural Netw. 18*(2), 551–572.

[7] Hill, K., Stange, G., & Mo, J. (1989). Temporal synchronization in the primary auditory response in the pigeon. *Hear. Res. 39*(1–2), 63–73.

[8] Indiveri, G., Chicca, E., & Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw. 17*(1), 211–221.

[9] Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., & Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci. 5*(73).

[10] Jeffress, L. A. (1948). A place theory of sound localization. *Journal of Comparative and Physiological Psychology 41*(1), 35–39.

[11] Konishi, M. (1993). Listening with two ears. *Sci. Am. 268*(4), 66–73.

[12] Lazzaro, J., & Mead, C. A. (1989). A silicon model of auditory localization. *Neural Comput. 1*(1), 47–57.

[13] Marder, E., & Goaillard, J.-M. (2006). Variability, compensation and homeostasis in neuron and network function. *Nat. Rev. Neurosci. 7*(7), 563–574.

[14] Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science 275*, 213–215.

[15] Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike-timing. *Biol. Cybern. 98*, 459–478.

[16] Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., Schmuker, M., Brüderle, D., Schemmel, J., & Meier, K. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci. 7*(11).

[17] Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., & Meier, K. (2012). Is a 4-bit synaptic weight resolution enough? - constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci. 6*(90).

[18] Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, pp. 1947–1950. IEEE Press.

[19] Schemmel, J., Grübl, A., Meier, K., & Müller, E. (2006). Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, pp. 1–6. IEEE Press.

[20] Senn, W., Schneider, M., & Ruf, B. (2002). Activity-dependent development of axonal and dendritic delays, or, why synaptic transmission should be unreliable. *Neural Comput. 14*(3), 583–619.

[21] Seo, J., Brezzo, B., Liu, Y., Parker, B., Esser, S., Montoye, R., Rajendran, B., Tierno, J., Chang, L., Modha, D., & Friedman, D. (2011). A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pp. 1–4.

IV A neuromorphic network for generic multivariate data classification

Michael Schmuker[1,2], Thomas Pfeil[3], and Martin Nawrot[1,2]

[1]*Institute for Biology, Freie Universität Berlin, Berlin, Germany*
[2]*Bernstein Center for Computational Neuroscience Berlin, Berlin, Germany*
[3]*Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany*

Abstract

Computational neuroscience has uncovered a number of computational principles employed by nervous systems. At the same time, neuromorphic hardware has matured to a state where fast silicon implementations of complex neural networks have become feasible. En route to future technical applications of neuromorphic computing the current challenge lies in the identification and implementation of functional brain algorithms. Taking inspiration from the olfactory system of insects we constructed a spiking neural network for the classification of multivariate data, a common problem in signal and data analysis. In this model, real-valued multivariate data is converted into spike trains using "virtual receptors" (VRs). Their output is processed by lateral inhibition and drives a winner-take-all circuit that supports supervised learning. VRs are conveniently implemented in software, while the lateral inhibition and classification stages run on accelerated neuromorphic hardware. When trained and tested on real-world data sets we find that the classification performance is on par with a Naive Bayes Classifier. An analysis of the network dynamics shows that stable decisions in output neuron populations are reached within less than 100 ms of biological time, matching the time-to-decision reported for the insect nervous system. Through leveraging a population code, the network tolerates the variability of neuronal transfer functions and trial-to-trial variation that is inevitably present on the hardware system. Our work provides a proof of principle for the successful implementation of a functional spiking neural network on a configurable neuromorphic hardware system that can readily be applied to real-world computing problems.

Introduction

The remarkable sensory and behavioral capabilities of all higher organisms are provided by the network of neurons in their nervous systems. The computing principles of the brain have inspired many powerful algorithms for data processing, most importantly the perceptron and, building on top of that, multilayer artificial neural networks, which are being applied with great success to various data analysis problems (1). While these networks operate with continuous values, computation in biological neuronal networks relies on the exchange of action potentials, or *spikes*.

Simulating networks of spiking neurons with software tools is computationally intensive, imposing limits to the duration of simulations and maximum network size. To overcome this limitation, several groups around the world have started to develop hardware realizations of spiking neuron models and neuronal networks (2-10) for studying the behavior of biological networks (11). The approach of the *Spikey* hardware system used in the present study is to enable high-throughput network simulations by speeding up computation by a factor of 10^4 compared to biological real time (12,13). It has been developed as a reconfigurable multi-neuron computing substrate supporting a wide range of network topologies (14).

In addition to providing faster tools for neurosimulation, high-throughput spiking network computation in hardware offers the possibility of using spiking networks to solve real-world computational problems. The massive parallelism is a potential advantage over conventional computing when processing large amounts of data in parallel. On the other hand, conventional algorithms are often difficult to implement using spiking networks for which many neuromorphic hardware substrates are designed. Novel algorithms have to be designed that embrace the inherent parallelism of a brain-like computing architecture.

A common problem in data analysis is classification of multivariate data. Many problems in artificial intelligence relate to classification in some way or the other, such as object recognition or decision making. It is the basis for data mining, and as such has widespread applications in industry. We interact with classification systems in many aspects of daily life, for example in the form of webshop recommendations, driver assistance systems, or when sending a letter with a handwritten address that is deciphered automatically in the post office.

In this work, we present a neuromorphic network for supervised classification of multivariate data. We implemented the spiking network part on a neuromorphic hardware system. Using a range of data sets, we demonstrate how the classifier network supports nonlinear separation through encoding by virtual receptors, while and lateral inhibition transforms the input data into a sparser encoding that is better suited for learning.

Results

We first outline our spiking neural network design and show examples of the network activity during operation in supervised classification of multivariate data. Then we analyze the temporal dynamics of the classification process and compare the network classification performance against the performance of a Naive Bayes classifier. We show that the network tolerates the neuronal variability that is present on the hardware through leveraging a population code. Finally, we demonstrate that the network design is generic and can be applied, without re-parameterization, to different multivariate

problems. We used the PyNN software package for network implementations on the *Spikey* neuromorphic hardware system (15,16). For simplicity, all temporal parameters are specified in the biological time domain throughout this study. The actual time values referring to the spiking network execution on the hardware are 10^4 times smaller due to the speed-up factor of the accelerated *Spikey* system.

A spiking network for supervised learning of data classification.

In multivariate classification problems, data is typically organized as observations of a number of variables arranged in a matrix $\mathbf{X}$, with rows corresponding to observations and columns to real-valued features. Each observation has an associated class label stored in a binary matrix $\mathbf{Y}$, with $Y_{i,j}=1$ if the observation i belongs to class j. The aim is to find a mapping $\mathbf{A}$ such that $argmax(\mathbf{X}\cdot\mathbf{A})=\mathbf{Y}$, with *argmax* returning 1 for the maximal value in each row and 0 otherwise. The classes of new observations $\mathbf{X'}$ can then be predicted by applying the transformation $argmax(\mathbf{X'}\cdot\mathbf{A})=\mathbf{Y'}$. The architecture of the insect olfactory system maps well on this task (17,18,19).

We designed a classifier network that approximates the basic blueprint of the insect olfactory system, without claiming to be an exact model of the biological reality. Its three-stage architecture consists of an input layer, a decorrelation layer and an association layer (Fig. 1A). We provide a detailed parameter list in the supplement (Table ST2 and supplemental Methods).

In the input layer, real-valued multidimensional data is transformed into bounded and positive firing rates. The data enters the network via ensembles of receptor neurons (RNs). RNs fire spikes at specified rates which are computed from the real-valued input data using *virtual receptors* (VRs) (17, see also supplemental Methods for details). A VR corresponds to the center of a linear (cone-shaped) radial basis function in feature space. The magnitude of its response to a data point (a *stimulus*) depends on the distance between the VR and the stimulus. Hence, the VR response is large for small distances between stimulus and receptor, and vice versa. VRs are placed in data space in a self-organized manner using the Neural Gas algorithm (20).

RN ensembles project onto projection neurons (PNs) in the decorrelation layer which are grouped in ensembles that represent the so-called glomeruli in the insect antennal lobe. Each RN ensemble targets one glomerulus, thus receives excitatory input that represents the activation of one VR. The PNs project to local inhibitory neurons (LNs), which laterally inhibit other glomeruli. Moderate lateral inhibition between glomeruli reduces correlations between the variables they represent without degrading the encoding to a fully orthogonalized representation (14,21,22,23).

The output of the decorrelation layer is projected to the association layer, in which supervised learning for data classification is realized. Association neurons (ANs) are grouped in as many populations as there are classes in the data set. Each population in the association layer is assigned one label from the data set (for example, “choice A” and “choice B” as indicated in Fig. 1A). The AN populations project onto associated populations of inhibitory neurons. The strong inhibition between AN populations induces a soft winner-take-all (sWTA) behavior in the association layer. The synaptic weights from PNs to ANs are initialized randomly. An activity pattern presented to the network will thus by chance deliver more input to one of the “choice” populations than to the others, resulting in higher firing rate of that population (the *winner population*). If the label of the winner population matches the one of the stimulus, the network performed a

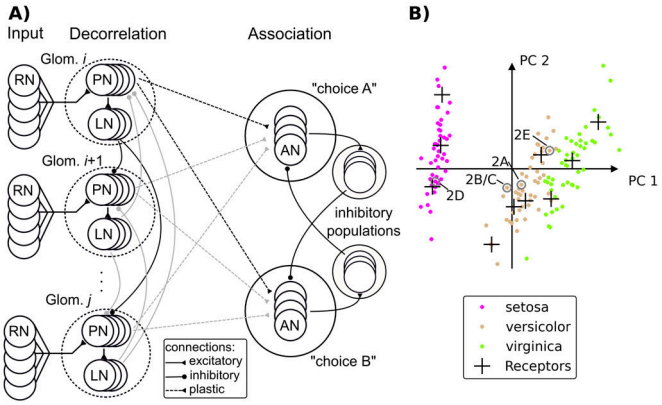


Fig. 1: Network architecture and real world classification problem. A) Schematic of the generic network. RN: Receptor neuron, PN: Projection neuron, LN: Local inhibitory neuron, AN: Association neuron. B) Projection of the complete Iris data set to the first two principal components (97.7% variance explained) and locations of ten VRs. Annotations refer to data points presented in Fig. 2.

correct classification. We used a 50% connection probability from RNs to PNs, from PNs to LNs and to ANs, and from excitatory to inhibitory neurons in the sWTA circuit (see supplementary Table ST2). Inhibitory populations are fully connected to excitatory populations.

We train the network in a supervised fashion by presenting stimuli with known class labels. If classification was correct, active synapses from PNs to the winner population are potentiated. If classification was incorrect, active synapses are depressed (see Methods for a detailed description of the algorithm). This learning rule is derived from the delta rule for perceptron training (24, 25). Network training leads to an optimized set of synaptic weights for classification of the data set. After successful training the winner population in the association layer indicates which class a stimulus belongs to, and it can predict the class adherence for unseen stimuli.

Application of the neuromorphic classifier network to a real-world data set.

We implemented the classifier network on the *Spikey* hardware system, which has been described in detail previously (14). We assessed its performance using Fisher's Iris data set (26) as a benchmark. The Iris data set is a four-dimensional data set describing features of the blossom leaves for three species of the Iris flower, *I. setosa*, *I. virginica* and *I. versicolor*. This data set is particularly well suited for this study for two reasons. First, it contains only 150 data points, which makes rapid prototyping of the network feasible. Second, the constellation of the data points allows for a fine-grained interpretation of the classifier capabilities: The *I. setosa* class is well separated from the other two, making learning the classification boundary easy (Fig. 1B). Separation of the *I. virginica* and *I. versicolor* classes is more difficult because they partly overlap in feature space. Classifier performance on this separation indicates how well the classifier copes with more challenging problems. Separating such overlapping data classes typically requires supervised learning methods, since there is no clear "gap" between the classes in data space that would allow an unsupervised method to detect class boundaries.

We used ten VRs to encode the data set. They represented the data points by firing intensities, which were used to generate the RN spike trains in the input layer using a

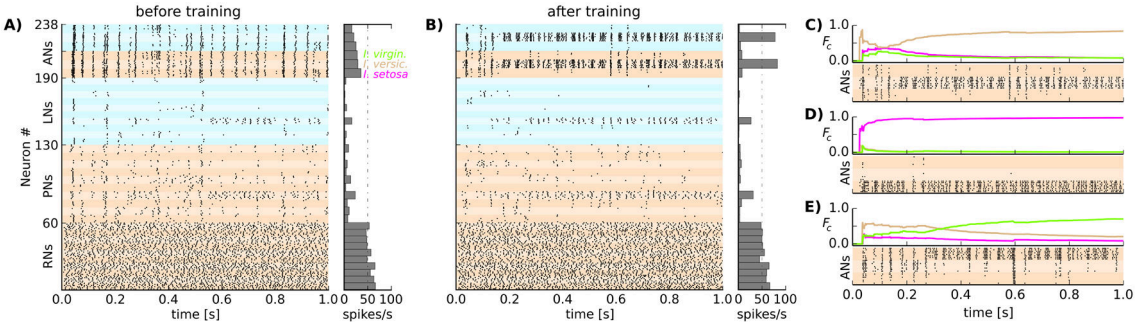


Fig. 2: Network activity during stimulus presentation before and after training. A) Untrained network. Spike raster display and population spike count of all neuron populations in the network in response to the presentation of one data point from *I. versicolor* as indicated in Fig. 1B. Distinct neuronal populations are labeled by alternating color saturation. Warm/cool color = excitatory/inhibitory population. The stimulus was applied at time t=0 s for the duration of 1 s. B) Network activity after training during 1 s of stimulation with a test sample from *I. versicolor* as labeled in Fig. 1B. C-E) Spiking activity and temporal evolution of $F_c(t)$ (eq. 1) for all three excitatory AN populations in response to three different data samples as labeled in Fig. 1B. Color of $F_c(t)$ trace indicates the *Iris* species associated to the respective AN population (color code as in Fig. 1B). Only spiking activity from excitatory ANs is shown.

gamma point process. The number of VRs determines the number of glomeruli, and thus the total number of neurons required for the network. The specific choice of ten VRs was a compromise between choosing a number as high as possible while staying within the maximal neuron count of 192 on the present neuromorphic hardware system (see supplemental Results for a detailed explanation).

The spiking activity of the classifier network is depicted in Fig. 2. Fig. 2A shows the activity of all neurons in all three layers in the beginning of the training phase when stimulated with the data point annotated as "2A" in Fig. 1B. The activity pattern across the RN population expresses the activation level of the VRs. PNs exhibited sparser activity compared to RNs, largely due to lateral inhibition from LNs. All three populations of ANs responded with approximately the same intensity since the weights from decorrelation layer to the association layer are initially random. Due to the strong lateral inhibition in the association layer, all three populations showed synchronized and oscillating activity. The population associated to the *I. setosa* class emitted a slightly higher number of spikes during the 1 s stimulus presentation than the others. Since the presented data point belonged to the *I. versicolor* class, this association was wrong, and hence the weights of synapses targeting the *I. setosa* population were reduced after this presentation as part of the training procedure. During the training phase, 80% of all data points were presented and the weights adjusted according to the learning rule after each presentation. Fig. 2B shows network activity in response to a sample from the *I. versicolor* class in the test phase. The AN population activity rapidly converged to a representation that indicated the correct association after only a few spikes and maintained this state throughout the duration of the stimulus presentation.

To assess the convergence of the association layer activity to a winner population, we calculated the cumulative fraction of spikes $F_c(t)$ from each population c at time t as

$$F_c(t) = \frac{I_c(t)}{I_{all}(t)} \quad , \qquad (1)$$

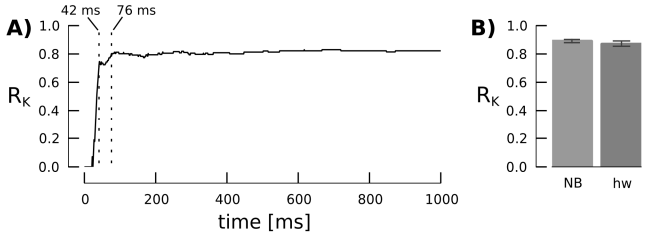


Fig. 3: Classification performance. A) R_K obtained for decision time points between 0 and 1000 ms from a single cross-validation run. Vertical dotted lines indicate when R_K first exceeds values of 0.7 (42 ms) and 0.8 (76 ms). B) Classification performance of the hardware classifier network (hw) at decision time of 1 s compared to a Naive Bayes classifier (NB) in fivefold cross-validation. Error bars: $20^{th}/80^{th}$ percentile from ten repetitions.

where $I_c(t)$ indicates the number of spikes emitted by population c within the interval (0,t], while $I_{all}(t)$ refers to the total number of spikes from all AN populations. $F_c(t)$ thus reflects, at each time point t, the integrated activity of one AN population compared to the total AN activity up to that point in time. Fig 2C shows the resulting population dynamics for the example in Fig. 2B, together with spike trains in the AN populations. For this data point, it took about 150 to 200 ms before the network activity converged towards a stable state with the *I.versicolor* population having the highest activity, indicating the correct association. This convergence happened faster for data samples from the well separated *I. setosa* class (Fig. 2D). In a third example from the *I.versicolor* class close to the class boundary, the network first showed a slightly higher firing rate for the correct class, but eventually converged to a wrong decision (Fig. 2E).

Time to decision and classification performance.

We used Gorodkin's K-category correlation coefficient R_K to measure classification performance (27, see eq. SE7 in the supplement). Compared to other frequently used performance measures like "percent correct", R_K is more sensitive to small performance differences when overall performance is already high and thus better suited for benchmarking. In addition, R_K is corrected for the bias introduced by skewed class proportions. For example, if 90% of the data are of one class and 10% the other class, we could yield "90% correct" classification by simply assigning all data samples to the first class. In contrast, R_K would report a value of zero, which is intuitively more accurate.

In our network each data class is represented by a different AN population. For each presentation of a test stimulus the population which generated the most spikes within a certain observation time window is the winner population, indicating either a correct or incorrect classification. We computed the R_K across all test samples in a time-resolved manner by varying the time t after stimulus onset that was used to count AN spikes. As shown in Fig. 3A, R_K rapidly approaches a stable maximum indicating a time-to-decision of less than 100 ms in biological time corresponding to 10µs of real-time with the *Spikey* chip.

We next compared the absolute classification performance with that of the Naive Bayes (NB) classifier, which we use here as a benchmark for conventional machine learning methods. We chose NB because it's a linear classifier without any free parameters, so it delivers robust classification without the need for parameter tuning. We evaluated the R_K across the entire 1 s stimulus presentation. For the Iris data set the NB classifier yields an average R_K value of 0.89 (P^{20}=0.88, P^{80}=0.90) in 50 repetitions of five-fold cross-validation and thus slightly outperforms the neuromorphic classifier with $R_K = 0.87$

(P^{20}=0.85, P^{80}=0.89, 50 repetitions) (Fig 3B). The performance evaluation is described in detail in the supplemental Methods.

For a thorough examination of the classification outcome we compared the confusion matrix produced by the classifier network (Table ST1, supplemental Results). The classifier only produced errors on the more challenging separation of *I. versicolor* and *I. virginica*, while it always succeeded to separate *I. setosa*. This observation indicates that the classifier network is capable of delivering reliable classification not only of well separable data, but also in cases where samples from different classes overlap in feature space.

Tolerance against neuronal variability.

The analog circuits used to represent neurons in the *Spikey* system exhibit inherent variability that the classifier network must tolerate in order to be useful in practice. Two sources of variability on the hardware system can be distinguished: "Temporal noise" and "fixed pattern noise". Temporal noise (including thermal noise and other sources of stochastic variability) affects the circuits on short timescales in an unpredictable fashion. In contrast, fixed pattern noise is caused by device mismatch. Device mismatch describes the deviance of an electronic component from its specification due to inevitable variations in the manufacturing process. The variations of neuron parameters due to device mismatch occur on much slower time scales and can be regarded as constant for our use case. They introduce heterogeneity across all analog components – neurons and synapses – according to a fixed pattern (hence the term "fixed pattern noise"). The individual variation can be measured and calibrated for. The integrated development environment of the *Spikey* system contains calibration methods that reduce the amount of fixed pattern variability. However, such generic calibration methods cannot account for all network configurations in an efficient manner, since calibration at the neuron level does not take into account network effects. This is particularly relevant for the *Spikey* system, which was designed to accommodate a wide variety of network topologies (14). In our case, the fixed pattern variation that remains after built-in calibration manifests itself in variability of the neurons' transfer functions that relate input rate to output rate. Both, maximal output rate and slope of the transfer functions varied considerably across PNs and LNs (Fig. 4A).

Due to its stochastic nature, temporal noise can't be avoided by systematic measures such as calibration of synaptic weights. We quantified the variation in spike count caused by temporal noise by measuring the variability of the spike count in all 192 hardware neurons across 50 repetitions with identical stimuli. For this purpose we generated input spike trains only once and used them repeatedly as input to all 192 neurons ("frozen input"). We used six gamma processes of order five and mean rate of 25 spikes/s to mimic the inputs that PNs receive in the classifier network. We adjusted the weights of the neurons to yield a mean output frequency of 25.4 spikes/s. The neurons exhibited moderate trial-to-trial variability under these conditions. Fig. 4B shows the distribution of spike counts for one exemplary neuron that produced 25.3 spikes on average, with a variance of 1.0. This amount of variability is also reflected when considering the total population (Fig. 4C). On average, the individual spike trains from the same neuron varied

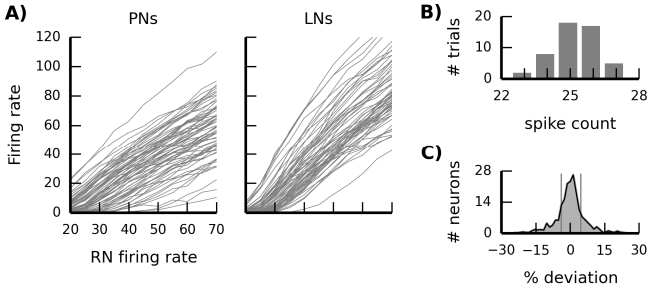


Fig. 4: Neuromorphic hardware variability. A) Variability of transfer functions across all PNs and LNs on the hardware chip. B) Hardware trial-to-trial variability: Histogram of spike counts for one example neuron across 50 repeated stimulations with identical "frozen input" of 1 s duration (average spike count: 25.3, variance 1.0). C) Histogram of per-neuron spike counts relative to the individual average spike count emitted by each of the 192 neurons across 50 repeated identical 1 s stimulations. Vertical lines: $20^{\text{th}}/80^{\text{th}}$ percentile (P^{20}=3.99, P^{80}=4.58).

with a Fano factor (28) of 0.083, which is smaller than the variability inherent to the gamma process used for the generation of the RN spike trains (γ=5, FF=0.2). Thus, the trial-to-trial variability due to temporal noise intrinsic to the neuromorphic hardware is small compared to those variations imposed by the biologically realistic stochastic generation of input spike trains.

The classifier network achieved the reported performance in spite of transfer function variability caused by fixed pattern noise and trial-to-trial variability caused by temporal noise and by the stochasticity of the input. This robustness is the result of considerable efforts to optimize network topology. Essentially, the key to achieve robustness in our network was to leverage population coding. Two network properties proved essential to ensure a valid population code. First, synchronization of neurons within a population should be avoided since it violates the rate code assumption of independent neurons within each population. We achieved this by sparsifying the input to individual neurons, i.e. using 50% connection probability instead of full connectivity. Second, population sizes must be sufficiently large in order to reduce the variance of the population transfer function. We provide a detailed explanation of how these properties affect network operation in the supplemental Results (Figs. S1, S2 and accompanying text "Network optimization for robustness against neuronal variability").

General applicability to other data sets.

As a demonstration for the ability of the network to solve nonlinear problems, we applied the network to classification of a two-dimensional "Ring" data set. This simple data set consists of two classes, one class situated in a cluster centered at the origin and a second class surrounding it (Fig. 5A). It has skewed class proportions with sevenfold more data points in the surround than in the center class. In addition, the arrangement of data points requires a nonlinear separation between the center and surround classes. Our network achieves this separation through the virtual receptor trick: By using ten VRs to represent a two-dimensional data set, we transform the data into a higher-dimensional space in which linear separation is possible. The classifier network running on the *Spikey* system achieved an average performance of R_K=0.96 on the Ring data set (Naive Bayes: R_K=0.98, Fig.5C, left).

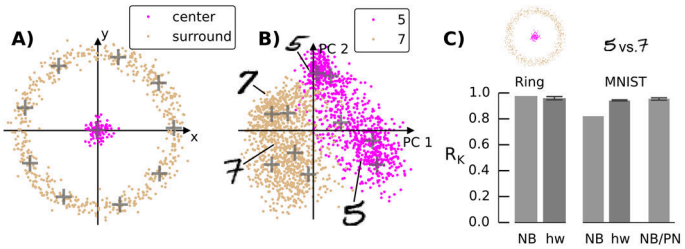


Fig. 5: Application to generic classification problems. A) Ring data set (training samples) and locations of ten VRs. B) PCA of MNIST digits '5' (893 samples) and '7' (1107 samples) from the training set (2000 samples total) and VR locations. C) Performance comparison of the Naive Bayes classifier (NB) vs. the classifier network on hardware (hw), and the NB classifier trained on PN firing rates (NB/PN). Error bars: $20^{\text{th}}/80^{\text{th}}$ percentile from ten repetitions. When trained on the VR responses, the NB performance is deterministic for these data sets because the training and test data sets are fixed. For NB/PN, we extracted the PN firing rates from the ten repeated network runs that we used to assess the spiking network. Hence, the NB classification performance varies.

The MNIST database is a commonly used high-dimensional benchmark problem with practical relevance (http://yann.lecun.com/exdb/mnist/). The database contains images of handwritten digits from 0 to 9, digitized to 28x28 pixels. Hence, each observation has $28 \cdot 28 = 768$ dimensions. The data set is divided into a training and a test set to enable reproducible benchmarking. We picked a subset of this data set consisting of the digits '5' and '7', using 2000 samples from the training set and 1920 samples from the test set (Fig. 5B). On the MNIST data set the spiking network outperformed the NB classifier by a large margin (hardware network: mean R_K=0.94, Naive Bayes: 0.82; Fig. 5C, right). Interestingly, when training the NB classifier on the spike counts produced by PNs in the network (that is, after the lateral inhibition stage in the decorrelation layer), its performance increases to similar levels as obtained with the classifier network (mean R_K=0.96). This observation is in line with a previous study which demonstrated that lateral inhibition increases classifier performance on a 184-dimensional odor data set (17).

The reason for this effect lies in the fact that lateral inhibition transforms the broad, overlapping receptive fields of VRs (and in consequence RNs) into more localized representations of input space. In other words, receptive fields of PNs are narrower than those of VRs, and they overlap to a lesser degree. As a result, PN activity is also sparser than VR activity, that is, only few PN populations respond to a particular stimulus. This behavior can be observed for example when comparing the spike counts of RNs and PNs in Fig. 2 A and B. Sparser activity and more local receptive fields simplify the training process, since it becomes easier to identify the input units that are relevant to discriminate data points in a particular region of input space (the "credit assignment problem"). We explain the soft partitioning effect provided by lateral inhibition in detail in the supplemental Results, including an illustrated example (Fig. S3).

Speed considerations.

The major advantage in using accelerated neuromorphic hardware for spiking neuronal simulations is its potentially fast execution time. On the neuromorphic hardware system used in this study simulations run with a speed-up factor of 10^4. Hence, presenting all 150 Iris data points for 1 s (biological time) each to the hardware network takes $150\text{s}/10^4$=15 ms pure network run time. Practical applications require data transfer for spikes and synaptic weights to and from the system as well as the parameterization of the hardware network, which adds to the pure network run time (for details see supplemental Methods and Fig. S4). These factors depend on the efficiency of the

software interface for the hardware system. Since we are working with a prototype setup, its interface is under constant development and improvement. At the time of writing, the hardware system effectively achieved an overall 13-fold speedup compared to biological real-time. We wish to stress that this number may improve as the software interface is continuously optimized.

Discussion

We demonstrated the implementation of a spiking neuronal network for classification of multidimensional data on a neuromorphic hardware system. The network is capable of separating data in a nonlinear fashion through encoding by virtual receptors (VRs). The transformation by lateral inhibition increases classification performance. It performed robustly in the presence of stochastic trial-to-trial variability inherent to the hardware system. The network is not restricted to any specific kind of data, but is capable of classifying arbitrary real-valued, multidimensional data, and hence universally suited for all kinds of classification tasks. It achieved performance values comparable to a standard machine learning classifier, which points out the network's wide applicability to real-world problems. The present network implementation is a proof of concept that can serve as a building block for classifier tasks on neuromorphic hardware. Together with the high speed-up factor of the neuromorphic hardware system, our universal classification network is an important step towards high-performance neurocomputing.

We verified the capability of our implementation of VRs to transform data into a higher-dimensional space in which linear separation is possible. The network we presented contains a linear classifier, with the additional constraint that the separating hyperplane must pass through the origin (29). As such, it is limited to separating linear problems. We overcame this limitation through the VR approach, which provides a higher-dimensional representation of the data. Our results on the MNIST data set point out that the lateral inhibition step is crucial for successful classification of real-world, high-dimensional data sets. While more complex machine learning algorithms like SVMs or Restricted Boltzmann Machines may allow for better classification performance directly on the VR data, the strength of our approach lies in the simplicity of a linear classifier combined with appropriate filtering of input data through the lateral inhibition step, that is very efficiently carried out in a massively parallel neuromorphic hardware network.

Lateral inhibition provides a soft partitioning of input space that facilitates classifier training. Note that this circumstance also points out a limitation of the presented classifier network, since class boundaries in data space can only be optimally represented if they coincide with partition borders. A straightforward way to deal with this problem is to increase the number of VRs and glomeruli, resulting in a more fine-grained partitioning of data space. Such an approach will be possible using emerging large-scale neuromorphic hardware systems supporting tens of thousands of neurons (7,30).

VRs depend on a self-organizing process that is trained in data space. A particularly interesting prospect is to implement this process on the neuronal substrate. Spiking self-organizing maps have been described in the literature (30,32,33), suggesting that, in principle, it is possible to implement a self-organizing process on a neuromorphic hardware system. However, the learning rules used in these studies would require sophisticated control logic, which makes it difficult to implement them on the *Spikey* system. A more straightforward and mathematically well founded approach has recently been put forward by Nessler and colleagues (34). They suggested a probabilistic, self-organizing mechanism to learn prototypes in feature space using spike-timing dependent

plasticity (STDP) and a winner-take-all circuit, which is suited to represent the VR encoding. An integrated implementation of this encoding together with the classifier network we present here will likely require a much higher neuron count and more flexible plasticity mechanisms compared to what is available on the *Spikey* system (13). On-chip implementations may become feasible considering the BrainScaleS wafer-scale hardware system that extends the number of available neurons by up to several orders of magnitude and provides more sophisticated plasticity mechanisms (6,35,36). In that system, multiple identical neuromorphic modules may be implemented on a single silicon wafer and communicate through high-bandwidth connections. Moreover, advanced control logic for on-chip implementation of elaborate STDP rules is under development (36), which is designed to be compatible with the self-organized prototype learning mechanisms described by Nessler et al. (34). In addition, the deterministic connectivity structure of the glomerular classifier network presented here facilitates splitting the network across different neuromorphic modules. The increased neuron count available in a large-scale system would allow for a larger number of VRs to solve more complex problems and enables scaling the network to larger population sizes in order to support robustness against noise.

Analysis of the dynamic network activation in response to the onset of a stimulus presentation revealed a fast decision time where the average performance reached its maximum within less than 100 ms in biological time (cf. Fig. 3A). This is in good agreement with recent measurements in insects. In the honeybee, a prominent animal model for studying learning and memory, it was shown that the encoding of the identity of an olfactory stimulus at the level of PNs evolved rapidly within tens of milliseconds (37,38). Neuronal populations at the output of the mushroom body encode odor-reward associations. These neuronal populations fulfill a similar function like the ANs in our network. In a classical conditioning paradigm, they indicated the classification of the conditioned stimulus (an odor that was previously paired with a sugar reward) within less than 200 ms (39).

Our network proved to be robust against neuronal variability, which is an important factor in the design of neuromorphic algorithms. Biological neuronal networks face a similar challenge. The study of neuronal variance is an integral part of today's neuroscience ever since the seminal study by Mainen and Sejnowski (40). Many neural properties are stochastic in nature, like neurotransmitter release or spike initiation, so a certain amount of variability is inevitable in biological neuronal networks (41). In the same vein, the analog nature of the circuits in the hardware enables the massive speed-up and integration density, but invariably entails variability. In our case, we achieved tolerance against variability by using a population code. Generally, accelerated analog neurocomputing requires models which can cope with and, ideally, make use of variability. The design of these models will benefit greatly from a deep understanding of biological circuits, interpreted in the light of variability. Likewise, creating functional networks on an analog neuromorphic substrate provides insight into critical properties that networks must possess in order to operate under noisy conditions.

Acknowledgements

We are deeply indebted to Daniel Bruederle for his support in using the *Spikey* system and we thank Eric Mueller for technical assistance and Mihai A. Petrovici and Emre Neftci for comments and discussion. We also thank the two anonymous reviewers for insightful comments that stimulated new experiments which led to great improvements of the manuscript. Author contributions: M.S. designed and performed research. T.P.

assisted in implementing the network on the *Spikey* hardware system. M.S., T.P. and M.N. wrote the manuscript. This work was supported by Deutsche Forschungsgemeinschaft (DFG grant no. SCHM2474/1-1 and 1-2 to M.S.), a grant from the German Ministry of Education and Research (BMBF), grant no. 01GQ1001D to M.S. (BCCN Berlin, Project A7) and M.N. (Project A9), and a grant from BMBF (Bernstein Focus Neuronal Learning: Insect Inspired Robots, grant no. 01GQ0941 to M.N.). Thomas Pfeil has received funding by the European Union 7th Framework Programme under grant agreement no. 243914 (Brain-i-Nets).

Materials and Methods

Network training and supervised learning rule.

Stimuli were presented to the classifier network in a sequential manner. For each stimulus i the corresponding feature vector $\mathbf{x}_i$ was obtained from the observation matrix $\mathbf{X}$, converted into a firing-rate presentation with VRs, from which spike trains were generated by a gamma point process (42). Each stimulus was presented for 1 s of biological time. Synaptic weights that fulfilled a Hebbian eligibility constraint were updated after each stimulus presentation. A synaptic weight was eligible for updating if the target neuron was a member of the winner population, and if the spike count emitted by the presynaptic neuron during the previous stimulus presentation exceeded a threshold (fixed to 35 spikes in the 1 s stimulus interval). Eligible synapses were potentiated by a fixed amount if classification was correct, or depressed by a fixed amount if classification was incorrect. A formal description of the training algorithm is available in the supplemental Methods.

Network training was implemented in an interactive chip-in-the-loop fashion: Stimuli were processed by the network on the chip. After each stimulus, the network response was evaluated on the host computer where the weight changes are calculated. The network was then re-configured and the next stimulus presented.

References

1. Hertz JA, Krogh AS, Palmer RG (1991) *Introduction to the theory of neural computation, Volume I* (Addison-Wesley).
2. Mahowald M, Douglas RJ (1991) A silicon neuron. *Nature* 354:515–518.
3. Indiveri G, Chicca E, Douglas R (2009) Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition *Cognit Comput* 1:119–127.
4. Yu T, Sejnowski TJ, Cauwenberghs G (2011) Biophysical neural spiking, bursting, and excitability dynamics in reconfigurable analog VLSI. *IEEE Trans Biomed Circuits Syst* 5:420–429.
5. Indiveri G *et al.* (2011) Neuromorphic silicon neuron circuits. *Front Neurosci* 5:73.
6. Renaud S *et al.* (2010) PAX: A mixed hardware/software simulation platform for spiking neural networks. *Neural Networks* 23(7):905–916.
7. Brüderle D *et al.* (2011) A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol Cybern* 104:263–296.
8. Furber S *et al.* (2012) Overview of the SpiNNaker System Architecture. *IEEE Trans Comput* 99:1-14.

9. Choudhary S *et al.* (2012) Silicon neurons that compute, in *Artificial Neural Networks and Machine Learning–ICANN 2012,* eds Villa AEP, Duch W, Érdi P, Masulli F, Palm G, (Springer Berlin Heidelberg), pp 121-128.
10. Merolla P *et al.* (2011) A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm, in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pp 1–4.
11. Imam N *et al.* (2012) Implementation of olfactory bulb glomerular-layer computations in a digital neurosynaptic core. *Front Neurosci* 6:83.
12. Brüderle D *et al.* (2010) Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system, in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (IEEE), pp 2784–2787.
13. Schemmel J, Gruebl A, Meier K, Mueller E (2006) Implementing synaptic plasticity in a VLSI spiking neural network model, in *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)* (IEEE Press, Vancouver), pp 1–6.
14. Pfeil T *et al.* (2013) Six networks on a universal neuromorphic computing substrate. *Front Neurosci* 7:11.
15. Davison AP *et al.* (2008) PyNN: A common interface for neuronal network simulators. *Front Neuroinform* 2:11.
16. Brüderle D *et al.* (2009) Establishing a novel modeling tool: a python-based interface for a neuromorphic hardware system. *Front Neuroinform* 3:17.
17. Schmuker M, Schneider G (2007) Processing and classification of chemical data inspired by insect olfaction. *Proc Natl Acad Sci USA* 104:20285–20289.
18. Huerta R, Nowotny T, Garcia-Sanchez M, Abarbanel HDI, Rabinovich MI (2004) Learning classification in the olfactory system of insects. *Neural Comput* 16:1601–1640.
19. Huerta R, Nowotny T (2009) Fast and robust learning by reinforcement signals: explorations in the insect brain. *Neural Comput* 21:2123–2151.
20. Martinetz T, Schulten K (1991) A "neural-gas" network learns topologies, in *Artificial Neural Networks,* eds Kohonen T, Mäkisara K, Simula O, Kangas J, (Elsevier B.V., North-Holland), pp 397–402.
21. Linster C, Sachse S, Galizia CG (2005) Computational modeling suggests that response properties rather than spatial position determine connectivity between olfactory glomeruli. *J Neurophysiol* 93:3410–3417.
22. Wick SD, Wiechert MT, Friedrich RW, Riecke H (2010) Pattern orthogonalization via channel decorrelation by adaptive networks. *J Comput Neurosci* 28:29–45.
23. Schmuker M, Yamagata N, Nawrot MP, Menzel R (2011) Parallel representation of stimulus identity and intensity in a dual pathway model inspired by the olfactory system of the honeybee. *Front Neuroeng* 4:17.
24. Brader JM, Senn W, Fusi S (2007) Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput* 19:2881–2912.
25. Soltani A, Wang X-J (2010) Synaptic computation underlying probabilistic inference. *Nat Neurosci* 13:112–119.
26. Fisher, R. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7(2):179–188.
27. Gorodkin J (2004) Comparing two K category assignments by a K-category correlation coefficient. *Comput Biol Chem* 28:367–374.

28. Nawrot MP, et al. (2008) Measurement of variability dynamics in cortical spike trains. *J Neurosci Meth* 169:374–390.
29. Häusler C, Nawrot MP, Schmuker M (2011) A spiking neuron classifier network with a deep architecture inspired by the olfactory system of the honeybee, in *Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on (IEEE)*, pp 198–202.
30. Schemmel J, *et al.* (2010) A wafer-scale neuromorphic hardware system for large-scale neural modeling, in *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)* (IEEE Press, Paris), pp 1947–1950.
31. Ruf B, Schmitt M (1998) Self-organization of spiking neurons using action potential timing. *Neural Networks, IEEE Transactions on* 9(3):575-578.
32. Choe Y, Miikkulainen R (1998) Self-organization and segmentation in a laterally connected orientation map of spiking neurons. *Neurocomputing* 21:139–158.
33. Behi T, Arous N (2011) Word Recognition in Continuous Speech and Speaker Independent by Means of Recurrent Self-organizing Spiking Neurons. *Signal Processing: An International Journal* 5(5):215–226.
34. Nessler B, Pfeiffer M, Buesing L, Maass W (2013) Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity. *PLoS Comput Biol* 9(4):e1003037.
35. Pfeil T, *et al.* (2012) Is a 4-bit synaptic weight resolution enough?–constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front Neurosci* 6:1.
36. Friedmann S, Frémaux N, Schemmel J, Gerstner W, Meier K (2013) Reward-based learning under hardware constraints - Using a RISC processor embedded in a neuromorphic substrate. *Front Neurosci* 7:160
37. Krofczik S, Menzel R, Nawrot MP (2008) Rapid odor processing in the honeybee antennal lobe network. *Front Comput Neurosci* 2:9.
38. Strube-Bloss MF, Herrera-Valdez MA, Smith BH (2012) Ensemble Response in Mushroom Body Output Neurons of the Honey Bee Outpaces Spatiotemporal Odor Processing Two Synapses Earlier in the Antennal Lobe. PLoS One 7:e50322. Available at: http://dx.plos.org/10.1371/journal.pone.0050322 [Accessed December 2, 2012].
39. Strube-Bloss MF, Nawrot MP, Menzel R (2011) Mushroom body output neurons encode odor–reward associations. *J Neurosci* 31(8):3129-3140.
40. Mainen ZF, Sejnowski TJ (1995) Reliability of spike timing in neocortical neurons. *Science* 268:1503–1506.
41. Boucsein C, Nawrot MP, Schnepel P, Aertsen A (2011) Beyond the cortical column: abundance and physiology of horizontal connections imply a strong role for inputs from the surround. *Front Neurosci* 5:32.
42. Tuckwell, HC (1988) *Introduction to theoretical neurobiology: Volume 2, nonlinear and stochastic theories* (Cambridge University Press).

Supplemental Results and Methods

Supplemental Results

Number of glomeruli

We used ten VRs and thus ten glomeruli in the network. The specific choice of ten was made as a compromise to use as many VRs as possible to encode the data while staying within the maximal neuron count of 192 on the present hardware system. In the following we describe this circumstance in more detail. First, each VR requires one glomerulus. One glomerulus consists of 6 input channels (RNs) and 13 neurons (7 PNs and 6 LNs). Ten glomeruli thus require 130 neurons and 60 additional synapse line drivers for input spikes (see (1) for a detailed technical explanation of the hardware system with regard to synapse line drivers). In addition, each AN population consists of 16 neurons (8 excitatory and 8 inhibitory neurons). For the Iris data set, we require three AN populations, or a total of 48 neurons. Since each neuron requires a synapse line driver, we thus require a total of 130 + 60 + 48 = 238 synapse line drivers. An additional VR would require 6 RNs + 7 PNs + 6 LNs = 19 additional line drivers, totaling to 257 and exceeding the maximum number of 256 synapse line drivers. Although it might be possible to gain space for one or two additional glomeruli by tuning the network to use fewer neurons per glomerulus, we don't expect a significantly different network behavior from a small increase in glomerulus number (see also (2) for an analysis of how VR count affects the performance of a Naive Bayes classifier). Large-scale neuromorphic hardware systems that are under development (e.g. (3,4)) will overcome this limitation and support thousands of neurons.

Per-class classification performance.

For a thorough examination of the classification outcome we depict the confusion matrix produced by the classifier network (Table ST1). The classifier only produced errors on the separation of *I. versicolor* and *I. virginica*, while it always succeeded to correctly separate *I. setosa* (R_K=0.87, P^{20}=0.85, P^{80}=0.89). *I. setosa* is well separated from the other classes in feature space. The classifier network achieved perfect separation in cross-validated training in all 50 repetitions. *I. versicolor* and *I. virginica* overlap in feature space, providing a harder challenge to the classifier that is reflected in the higher error rate for that particular separation.

Network optimization for robustness against neuronal variability.

Constructing a heterogeneous network under constraints of limited neuron count and bounded synaptic weights imposes a tradeoff in connectivity: The number of neurons in a population that project on postsynaptic neurons (the postsynaptic "fan-in") must be sufficiently large to be able to drive the postsynaptic neuron to spiking, but the population must be kept small in order to accommodate many populations. In a previous version of the network, we achieved the maximum possible postsynaptic fan-in by using all-to-all connectivity (connection probability p_{conn}=100%) between all connected populations. In addition, that network contained only three LNs per glomerulus (instead of six in the current, optimized network). Since the fan-in of LNs on PNs is large anyway, this decision seemed a viable way to reduce the total neuron count. Achieving good classification performance with that network required a network-specific calibration routine (see supplemental Methods). The calibration improved the homogeneity of the

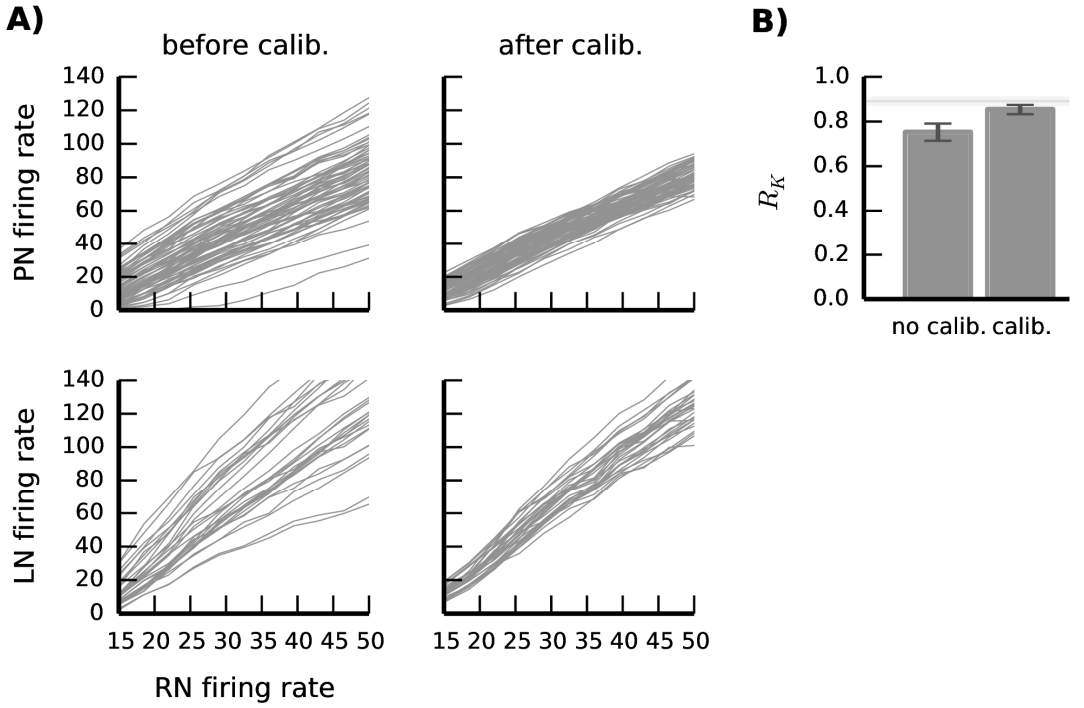


Fig. S1: Neuronal variability on the hardware system and impact of calibration on classifier performance using a previous version of the network with 100% connectivity. A) Rate-response functions of the hardware neurons, before (left) and after (right) calibration (5 s stimulation duration). Upper row: PNs, lower row: LNs. B) Classifier performance in the Iris benchmark before and after network specific calibration. Error bars denote P^{20} and P^{80}. Horizontal gray bar: Naive Bayes performance.

transfer functions (Fig. S1A) and classification performance improved from R_K around 0.75 to values around 0.86 (Fig. S1B).

Analyzing the operation of the fully connected network in detail, we found that neurons in a population were highly synchronized (Fig. S2). This synchronization at the population level was a direct consequence of full connectivity, which entails that all neurons of a population receive the same input. For example, all PNs in a glomerulus received input from the same set of RNs, and their spiking activity was highly correlated as a consequence of the common input. The same was true for LNs, and populations on the AN level. Under these conditions, the assumption of a population rate code with independent neurons is violated – the whole population of n neurons acts like a single neuron with n times the synaptic weight. Clearly, the postsynaptic interaction of excitatory and inhibitory inputs in this regime is impaired, since synchronous PN spikes lead to synchronized inhibitory LN spikes with a short delay. In contrast, if all n neurons fire independently the post-synaptic cell receives the same total number of spikes, but distributed more evenly in time. Hence, the chance that excitatory and inhibitory post-synaptic potentials overlap is considerably higher in the asynchronous case. In the optimized network we reduced the synchronization within relevant populations by sparsifying the connectivity from RNs to PNs and from PNs to LNs and ANs by 50% and re-adjusting the synaptic weights accordingly.

As an additional step to increase the robustness against transfer function variability, we increased the number of LNs from three to six. Since the individual transfer functions of LNs underlie variability on the hardware, the total transfer function of an LN population will vary according to σ^2/n, with n the population size and σ^2 the variance of the

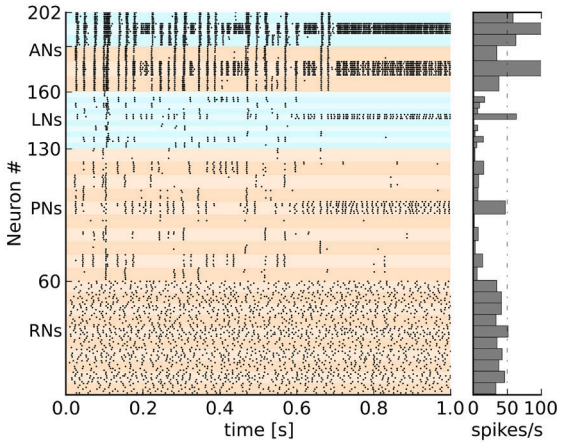


Fig. S2: Synchronized spiking activity in PNs, LNs and ANs in a previous version of the network with p_{conn}=100%. The total neuron count in the previous network is lower than in the version presented in the main manuscript due to different per-population neuron counts for LNs (3 in the previous network vs. 6 in the main manuscript) and inhibitory ANs (6 vs. 8).

individual transfer functions. Thus, increasing LN population size decreases variability of LN population transfer functions. In consequence, the inhibition strength that a PN population receives from other glomeruli becomes more homogeneous. In other words, increasing LN population size decreases the likelihood that a particular glomerulus may exert significantly higher inhibition than the others and thus alleviates the impact of transfer function variability.

Taken together, we achieved robustness to transfer function variability by two measures: First, we improved population rate coding by making the connectivity sparser, thus alleviating strong coupling on the postsynaptic side. Second, we increased the size of LN populations, thus reducing the variance of the population transfer functions of the LN groups. These steps resulted in the present network that is robust against variability in the transfer functions of individual neurons.

Effect of lateral inhibition on classification performance

The result that the Naive Bayes classifier's performance increases if trained on the PN firing rates compared to training on the VR responses (Fig. 5C in the main text) points out the beneficial effect of lateral inhibition in the presented network. Lateral inhibition transforms the broad, overlapping receptive fields of VRs into localized and more selective receptive fields on the PN level. This step facilitates the "credit assignment problem", that is, the identification of the PNs (or more precisely the PN-AN synapses) that are most responsible for the classification outcome. This information is necessary to select the correct synapses to be potentiated or depressed during classifier training (the "credit assignment problem").

Fig. S3 shows a sketch to illustrate this circumstance. Consider the VR "R_2" in Fig. S3A. Since the distances d_1 and d_2 are equal, the response of R_2 to the respective points will be equal, because it depends linearly on these distances (see eq. SE1). Thus, the response magnitude of this particular VR provides ambiguous information with regard to class adherence, which complicates the learning process. Moreover, since VR receptive fields are broad, there is considerable overlap in the receptive fields of R_1 and R_2 (Fig. S3B). One could now simply reduce the receptive field size of VRs (Fig. S3C). However, this approach would cause many data points not to be covered by any receptive field – the

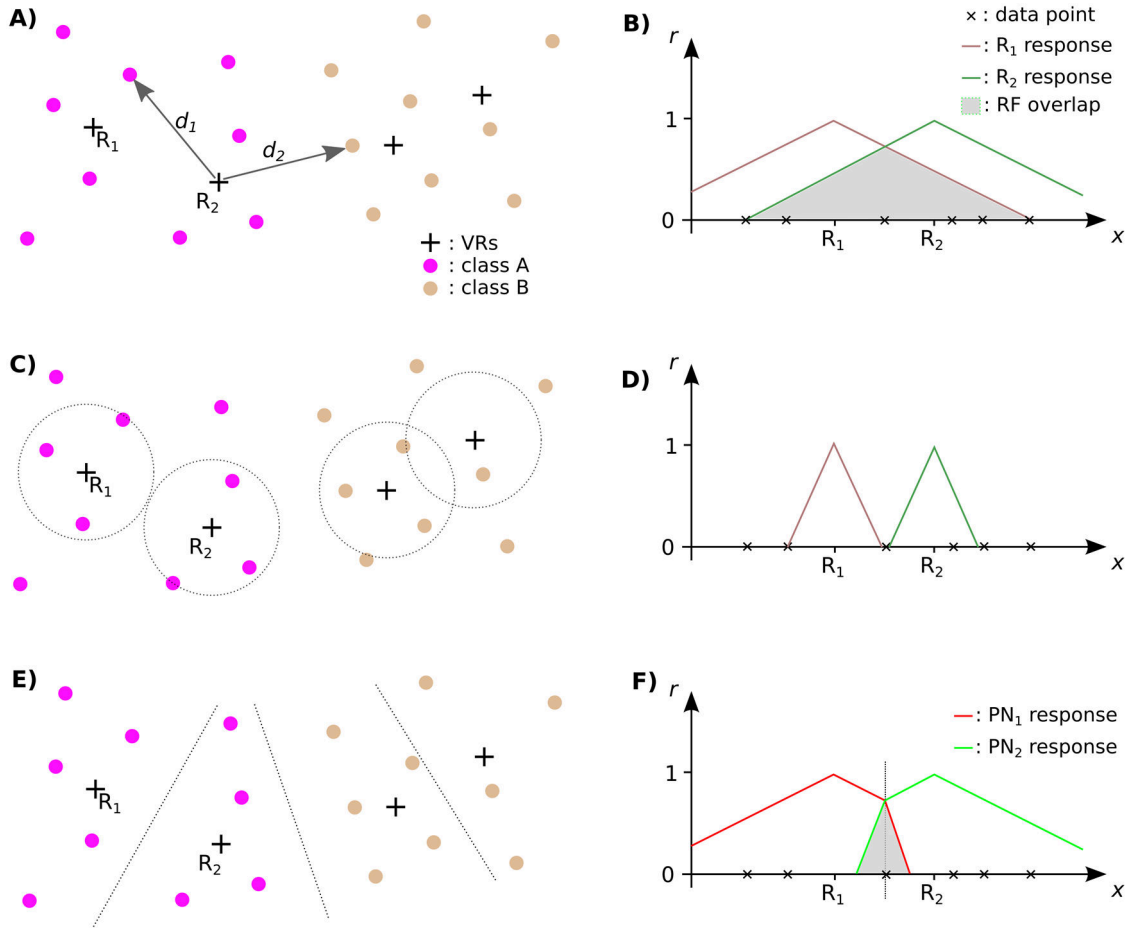


Fig. S3: Illustration of the credit assignment problem and the effect of lateral inhibition. A) Cartoon of a hypothetical two-class, two-dimensional classification problem with VRs. The distances d_1 and d_2 are equal. B) One-dimensional sketch of the response profile of the two VRs R_1 and R_2. C) Effect of reducing VR receptive field size in data space. D) Effect of reducing VR receptive field size on the response profiles. E) Voronoi partitioning of input space with VRs as generators. F) Effect of lateral inhibition on PN receptive field size.

network would be "blind" towards these data points (Fig. S3D). They could neither be used for training, nor could the trained network achieve correct classification to any data point in the "blind" areas. Moreover, as the density of VRs in different regions of data space may be different, choosing one RF size for all VRs is clearly not optimal.

Lateral inhibition solves this problem in an elegant way: The response of a VR to a data point will be attenuated by lateral inhibition on the PN level if another VR is closer. Every PN thus has an "authoritative" region in data space where it provides the highest response and the responses of PNs in other glomeruli are attenuated. This region is equivalent to the Voronoi partitioning of input space with the VRs as generators (symbolized by the dotted lines in Fig. S3E). The resulting PN receptive fields become narrower in regions where there is overlap, but retain their full extent in regions where no other PN competes (Fig. S3F). Hence, lateral inhibition between PNs optimally and efficiently partitions data space on the PN level. Each PN thus represents a region in input space for which it is authoritative, considerably simplifying the credit assignment problem.

Why does the Naive Bayes classifier benefit from lateral inhibition? This classifier estimates the mean μ and the variance σ^2 of each class along each dimension in its input space. Classification is then achieved by comparing the (naively) estimated probability of adherence to class 1 vs. class 2. These probabilities are computed from the multivariate normal distributions $N(\mu,\sigma^2)$, with μ and σ^2 the means and variances along each dimension of input space. Broad VR receptive fields entail high variance of VR responses, thus the estimated variance of the multivariate response distribution will also be high. In contrast, PN responses exhibit smaller variance since their receptive fields are narrower. Thus, the estimated variance of the PN response distribution will be smaller, and in consequence the Naive Bayes estimate of class adherence will exhibit lower variance, allowing for a better discrimination of classes in data space.

Supplemental Methods

Network parameters.

Each glomerulus was driven by six RNs, and contained seven PNs and six LNs. Each population in the associative layer comprised eight excitatory and eight inhibitory neurons. Connectivity and synaptic weights are described in detail Table ST2 (table design from (5)). For a schematic overview of the general network architecture see Fig. 1A in the body of the main manuscript. Time constants in the table refer to the biological value they model. The actual values on the hardware are 10^4 times smaller, due to the 10^4 speedup factor at which the hardware operates (6,7). The weights are specified as fractions of the maximal weight $w_{\max}^{\mathrm{hw}\,\{\mathrm{inh,exc}\}}$ for excitatory and inhibitory synapses in the hardware system, where $w_{\max}^{\mathrm{hw\,inh}} \sim 4 \cdot w_{\max}^{\mathrm{hw\,exc}}$. Neurons were implemented as standard integrate-and-fire models (see (1) for details).

Virtual receptors.

The response r of a virtual receptor with coordinates $\mathbf{p}$ to the stimulus $\mathbf{s}$ is given by eq. SE1,

$$r = 1 - \frac{d(\mathbf{s},\mathbf{p}) - d_{\min}}{d_{\max} - d_{\min}} \quad , \qquad \text{(SE1)}$$

with $d(\mathbf{s},\mathbf{p})$ the Manhattan distance (Minkowski metric with k=1, sum of absolute coordinate differences) between $\mathbf{s}$ and $\mathbf{p}$; $d_{\min}$ and $d_{\max}$ denote the minimum and maximum distance observed in the data set. Hence, the receptor response is a value in $[0, 1]$, and it is inversely proportional to the distance between stimulus and receptor.

The receptive fields implemented by eq. SE1 are equivalent to linear radial basis functions (RBFs) representing cones. They extend over the entire space that is covered by the data ("broadly tuned"). Their receptive fields are largely overlapping. This guarantees that there are no ‘‘blind spots" in data space that are not covered by any receptive field.

Virtual receptors were placed in data space using a self-organizing process. In this study, we used the neural gas algorithm (8), as implemented in the MDP toolkit (9). The neural gas learns to represent the distribution of data in the original coordinate space, thus ensuring that the virtual receptors cover data space appropriately. Each node in the neuronal gas corresponds to one virtual receptor. Using n virtual receptors, a stimulus

will thus evoke a response vector $\mathbf{r} = (r_1, \ldots, r_n)$. The elements of response vector r_i are then converted into firing rates ρ_i using eq. SE2,

$$\rho_i = r_i \cdot (\rho_{\max} - \rho_{\min}) + \rho_{\min} \quad \text{for} \quad \mathrm{i} = (1, \ldots, n)\,, \tag{SE2}$$

with $\rho_{\min}$ and $\rho_{\max}$ the minimal and maximal firing rate, set to 20 and 70 spikes/s, respectively. Firing rates were transformed into spike trains using a Gamma process of order five. The waiting time between stimulus onset and the first RN spike was drawn from the appropriate waiting time distribution, in our case a gamma distribution of order six, in order to prevent synchronization of RNs at stimulus onset. We chose a gamma process to generate spike times because its spiking statistics compares realistically to biological neurons (see e.g. (10)). In addition, the increased regularity of a gamma process of order five (Fano factor FF=0.2) compared to a Poisson process (FF=1.0) reduces the spike count variability and thus yields a more reliable encoding of input firing rates.

VRs were implemented in software as a convenient approach to convert numerical data into a spiking format. The VR approach satisfies the need for dimensionality reduction due to limited neuron counts and the need for a generic approach to convert real-valued data into bounded firing rate intervals.

Network training and supervised learning rule.

The classifier network was trained using a supervised learning algorithm. Only synapses between PNs and excitatory association layer neurons were subject to learning.

After stimulus presentation, a synapse was eligible for weight update if it fulfilled a Hebbian eligibility constraint. A synaptic weight was eligible for updating if the target neuron υ_{target} was a member of the winner population Υ_{winner}, and if the firing rate ρ_{pre} of the presynaptic neuron during the previous stimulus presentation exceeded a threshold θ (fixed to 35 spikes/s in this study). The eligibility constraint ε can thus be formalized as

$$\varepsilon = \begin{cases} 1, & \text{if } \rho_{\text{pre}} > \theta \text{ and } \upsilon_{\text{target}} \in \Upsilon_{\text{winner}}, \\ 0, & \text{otherwise.} \end{cases} \tag{SE3}$$

The change of the weight $\Delta w_{\text{PN}\to\upsilon}$ between any PN and target neuron υ in the association layer was governed by eq. SE4,

$$\Delta w_{\text{PN}\to\upsilon} = \begin{cases} \varepsilon \cdot \mathrm{c}, & \text{if classification was correct,} \\ -\varepsilon \cdot \mathrm{c}, & \text{if classification was incorrect,} \end{cases} \tag{SE4}$$

with c a constant value determined by the granularity of synaptic weights on the hardware (1). The new weight w_{new} was computed from w_{old} as in SE5,

$$w_{\text{new}} = w_{\text{old}} + \Delta w_{\text{PN}\to\upsilon}\ . \tag{SE5}$$

Synaptic weights were bounded in the interval $[w_{\min}, w_{\max}]$ by the constraints of the hardware. Thus, the final value of the synaptic weight was given by eq. SE6,

$$w_{\text{final}} = \begin{cases} w_{\max}, & \text{if } w_{\text{new}} > w_{\max}, \\ w_{\min}, & \text{if } w_{\text{new}} < w_{\min}, \\ w_{\text{new}}, & \text{otherwise.} \end{cases} \tag{SE6}$$

Evaluation of Classifier performance.

Classifier performance was evaluated from fivefold cross-validation (CV). The data was split into five equal parts, and four parts were used in training and one part was used to test the classifier predictions in each CV run. After five runs, each data point was once in the test set, allowing computing a single performance value for all five CV runs. CV was repeated multiple times with different random splitting of the data into five equal parts.

Classifier performance (i.e., prediction accuracy) was assessed using Gorodkin's R_K correlation coefficient for discrete multi-category data (11). The aim is to compare a prediction $\mathbf{Y}_{\text{pred}}$ to the true target values $\mathbf{Y}$, with $Y_{n,k} \in \{0,1\}$ for n predictions of k classes. The $K \times K$ confusion matrix $\mathbf{C}$ contains the number of correctly and falsely predicted data instances per class. $C_{k,k}$ contains the number of correctly predicted instances of class k, and off-diagonal elements contain the number of falsely predicted instances. For example, $C_{1,2}$ contains the number of instances predicted to belong to class 1, but actually belonging to class 2. The K-category correlation coefficient computes as in eq. SE7.

$$R_K = \frac{\sum_{klm} C_{k,k} C_{l,m} - C_{k,l} C_{m,k}}{\sqrt{\sum_k (\sum_l C_{k,l})(\sum_{l',k' \neq k} C_{k',l'})} \sqrt{\sum_k (\sum_l C_{l,k})(\sum_{l',k' \neq k} C_{k',l'})}} \quad . \tag{SE7}$$

Application-specific calibration of the neuromorphic hardware system.

The network-specific calibration for the previous version of the network with 100% connectivity (see supplemental results above) consisted of two steps. We first calibrated the PNs for homogeneous rate response, before calibrating the LNs. Calibration was carried out with the weight of all inhibitory synapses set to zero. We first measured PN firing rates in response to a one-second stimulation with maximum intensity, formed the median from all PN rates and used this as target firing rate. The "fitness" of the rate distribution was assessed by mean square deviation (MSD) of PN firing rates from the targeted PN firing rate,

$$\text{MSD} = \frac{1}{n} \sum_1^n \left(\rho_{\text{goal}}^{\text{PN}} - \rho_i^{\text{PN}} \right)^2 \quad , \tag{SE8}$$

with n the number of PNs, ρ_i^{PN} the firing rate of the i^{th} PN, and $\rho_{\text{goal}}^{\text{PN}}$ the targeted firing rate. The weights w_i from the RNs to the i^{th} PN were then updated according to

$$w_i^{\text{new}} = w_i \cdot \frac{\rho_{\text{goal}}^{\text{PN}}}{\rho_i^{\text{PN}}} \, . \tag{SE9}$$

In this case we relied on the automatic conversion of the *Spikey* control software that mapped the weight values into the discrete distribution required by the hardware (1).

When the MSD failed to decrease over five iterations, optimization was terminated and the weight set that yielded the best MSD until then was used. After the weights from RNs to PNs were optimized, we adjusted the weights between PNs and LNs using the same algorithm.

Speed considerations for the neuromorphic hardware system

The execution of the network on the accelerated hardware happens extremely fast: A simulation lasting for 150 s biological time is executed in 15 ms (a 10^4 speedup factor).

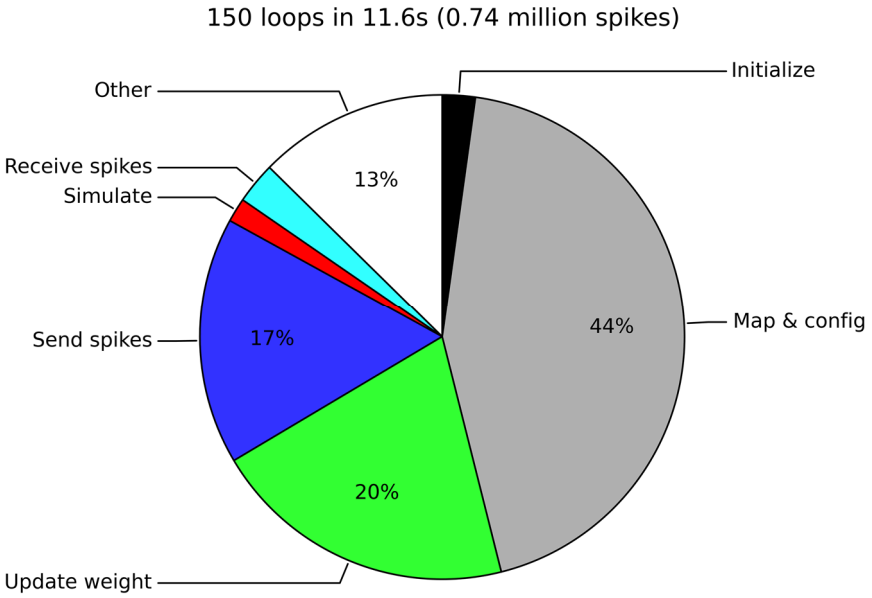


Fig. S4: Simulation time for one crossvalidation run (150 simulations) broken down into discrete steps. The largest fraction of the time is required by mapping the simulation parameters to hardware-compatible values and configuring the hardware network. Some of these tasks have to be repeated for every simulation, adding up to a substantial amount of total time. The second largest chunk is taken up by updating weights. The actual simulation requires less than 2% of the total time. "Other" encompasses numerous small tasks like handling of spike data and network configuration in the PyNN interface code. All numbers are subject to change as the software interface evolves.

However, the total run time of the classifier network is mainly determined by other factors, which we describe in the following.

A typical crossvalidation run requires 150 stimulus presentations of 1 s duration. Before starting such a simulation session, generic calibration data must be loaded and applied. The network connectivity as well as synaptic weights must be encoded and transferred, and subsequently be mapped from their specification in biologically realistic physical units to the appropriate hardware parameters. In addition, for each of the 150 simulations, spike data needs to be sent to and received from the hardware, including transfer, encoding and decoding of spike times and neuron IDs. During the training phase of the classifier synaptic weights also have to be updated before every stimulus presentation.

The absolute duration of these additional factors depends heavily on the efficiency of the software interface that links the hardware with the host system. Since it is a prototype system, this software interface is constantly developed and improved. It is therefore difficult to state an absolute number for the effective speedup achieved by offloading network simulations to the hardware. In order to give the reader the opportunity of an informed estimate, we analyzed how much time is required by each of the above steps (Fig. S4).

Several of these steps still bear potential for optimization. For example, the time required for weight update could be drastically shortened by differential configuration, i.e. updating only those hardware weights which have changed, instead of overwriting all weights as in the current implementation. In addition, on the current system all spike times produced in the network are being transferred back to the host system during

training and testing phases of the classifier network. The interface can be improved to only transfer those spikes which are necessary for the off-chip calculation of the weight change, namely PNs and excitatory ANs, and not transferring spike times from LNs and inhibitory ANs. When the network is completely trained, only the spike times from excitatory ANs are needed, further reducing the overhead due to handling spike data. We plan to implement these optimizations in future versions of the software interface.

Supplemental References

1. Pfeil T et al. (2013) Six networks on a universal neuromorphic computing substrate. *Front Neurosci* 7:11.
2. Schmuker M, Schneider G (2007) Processing and classification of chemical data inspired by insect olfaction. *Proc Natl Acad Sci USA* 104:20285–20289.
3. Furber S *et al.* (2012) Overview of the SpiNNaker System Architecture. *IEEE Trans Comput* 99:1-14.
4. Schemmel J, *et al.* (2010) A wafer-scale neuromorphic hardware system for large-scale neural modeling, in *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)* (IEEE Press, Paris), pp 1947–1950.
5. Nordlie E, Gewaltig M-O, Plesser HE (2009) Towards Reproducible Descriptions of Neuronal Network Models. *PLoS Comput Biol* 5:e1000456.
6. Brüderle D et al. (2010) in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (IEEE)*, pages 2784–2787.
7. Schemmel J, Gruebl A, Meier K, Mueller E (2006) in *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)* (IEEE Press, Vancouver), pages 1–6.
8. Martinetz T, Schulten K (1991) in *Artificial Neural Networks*, eds Kohonen T, Mäkisara K, Simula O, Kangas J (Elsevier B.V., North-Holland), pages 397–402.
9. Zito T, Wilbert N, Wiskott L, Berkes P (2008) Modular Toolkit for Data Processing (MDP): A Python Data Processing Framework. *Front Neuroinform* 2:8.
10. Nawrot MP et al. (2008) Measurement of variability dynamics in cortical spike trains. *J Neurosci Methods* 169:374–390.
11. Gorodkin J (2004) Comparing two K-category assignments by a K-category correlation coefficient. *Comp Biol Chem* 28:367–374.

Table ST1: Average count of predicted vs. actual class adherence (columns vs. rows) obtained across 50 repetitions of fivefold cross-validation.

	I. setosa	*I. versicolor*	*I. virginica*
I. setosa	50.0	0.0	0.0
I. versicolor	0.0	47.1	10.7
I. virginica	0.0	2.9	39.3

Table ST2: Network parameters.

Receptor Neurons (RNs)	
Type	Gamma process (γ=5)
Count	6 RNs per virtual receptor
Outgoing connectivity	Each RN projects on the PNs in one glomerulus, connection probability p_{conn}=50%
Outgoing weights	RN to PN: $0.5 \cdot w_{\text{max}}^{\text{hw exc}}$
Projection Neurons (PNs)	
Type	Leaky integrate-and-fire
Count	7 PNs per glomerulus
Outgoing connectivity	Excitatory synapses on LNs in the same glomerulus (p_{conn}=50%) and on excitatory ANs (p_{conn}=50%)
Outgoing weights	PN to LN: $0.7 \cdot w_{\text{max}}^{\text{hw exc}}$. PN to AN: initially random between $0.2 \cdot w_{\text{max}}^{\text{hw exc}}$ and $0.66 \cdot w_{\text{max}}^{\text{hw exc}}$ (adjusted in training).
Local inhibitory Neurons (LNs)	
Type	Leaky integrate-and-fire
Count	6 LNs per glomerulus
Outgoing connectivity	Inhibitory synapses on all PNs in all other glomeruli (p_{conn}=100%)
Outgoing weights	LN to PNs: $0.133 \cdot w_{\text{max}}^{\text{hw inh}}$
Excitatory neurons in association layer (ANs)	
Type	Leaky integrate-and-fire
Count	8 per association population
Outgoing connectivity	Excitatory synapses on adjoint inhibitory population (p_{conn}=50%)
Outgoing weights	AN to adjoint inhibibitory population: $0.5 \cdot w_{\text{max}}^{\text{hw exc}}$
Inhibitory neurons in association layer	
Type	Leaky integrate-and-fire
Count	8 per association population
Outgoing connectivity	Inhibitory synapses on excitatory neurons of all other association populations (p_{conn}=100%).
Outgoing weights	Inhibitory neuron to ANs different association populations: $1.0 \cdot w_{\text{max}}^{\text{hw inh}}$

V The effect of heterogeneity on decorrelation mechanisms in spiking neural networks: a neuromorphic-hardware study

Thomas Pfeil[1,†], Jakob Jordan[2,†], Tom Tetzlaff[2], Andreas Grübl[1], Johannes Schemmel[1], Markus Diesmann[2], and Karlheinz Meier[1]

[1]*Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany*
[2]*Inst. of Neuroscience and Medicine (INM-6), Computational and Systems Neuroscience & Inst. for Advanced Simulation (IAS-6), Theoretical Neuroscience, Jülich Research Centre and JARA, Jülich, Germany*

Abstract

Correlations in neural activity can severely impair the processing of information in neural networks. In finite-size networks, correlations are however inevitable due to common presynaptic sources. Recent theoretical studies have shown that inhibitory feedback, abundant in biological neural networks, can actively suppress these shared-input correlations and thereby enable neurons to fire nearly independently. For networks of spiking neurons, the decorrelating effect of inhibitory feedback has so far been explicitly demonstrated only for homogeneous networks of neurons with linear sub-threshold dynamics. Theory, however, suggests that the effect is a general phenomenon, present in any system with inhibitory feedback, irrespective of the details of the network structure and the neuron and synapse properties. Here, we investigate the effect of network heterogeneity on correlations in sparse, random networks of inhibitory neurons with conductance-based synapses. Accelerated neuromorphic hardware is used as a user-friendly stand-alone research tool to emulate these networks. The configurability of the hardware substrate enables us to modulate the extent of network heterogeneity in a systematic manner. We selectively study the effects of shared-input and recurrent connections on correlations in synaptic inputs and spike trains. Our results confirm that shared-input correlations are actively suppressed by inhibitory feedback also in highly heterogeneous networks exhibiting broad, heavy-tailed firing-rate distributions. However, while cell and synapse heterogeneities lead to a reduction of shared-input correlations (feedforward decorrelation), feedback decorrelation is impaired as a consequence of diminished effective feedback.

[†]These authors contributed equally to this study.

1 Introduction

Spatial correlations in the activity can severely impair information processing in neural networks [1, 2, 3, 4]. A functional benefit of small cross-correlations has been demonstrated, e.g., in [5], showing that decreased spike-train correlations are accompanied by increased task performance. In finite-size neural networks, an inevitable source of correlated neural activity is presynaptic input shared by multiple postsynaptic neurons. In network models and in-vivo recordings, however, pairwise correlations in the activity of neighboring neurons have been found to be much smaller than expected given the amount of shared input [6, 7, 8, 9, 10, 11, 12]. Renart et al. [13] and Tetzlaff et al. [14] have attributed this observation to an active decorrelation of neural activity by negative feedback. Negative feedback, abundant in biological neural networks, can effectively suppress pairwise correlations and fluctuations in the population activity, and thereby enable neurons to fire nearly independently despite substantial shared input.

For networks of spiking neurons, decorrelation by inhibitory feedback has so far been explicitly demonstrated only for the homogeneous case, where all neurons have identical properties, receive (approximately) the same number of inputs, and, hence, fire at about the same rate [13, 11]. Moreover, the sub-threshold dynamics of individual neurons was assumed to be linear. The underlying theory [14], however, suggests the effect to be much more general: Decorrelation should be observable in any system with sufficiently strong inhibitory feedback, irrespective of the details of the network structure and the cell and synapse properties.

Biological neuronal networks often exhibit broad, heavy-tailed firing-rate distributions [15, 16, 17, 18, 19, 20, 21], indicating a high degree of heterogeneity, e.g., in synaptic weights [22, 23, 24, 25, 26], in-degrees [27] or time constants [28, 20]. The same holds for neural networks implemented on analog neuromorphic hardware: Analog neuromorphic hardware substrates [29] mimic properties of biological nervous systems using physical models of neurons and synapses [30] (capacitors, for example, emulate insulating cell membranes). In consequence, neural-network emulations on analog neuromorphic hardware are massively parallel, energy efficient and potentially extremely fast, thereby making these substrates highly attractive as tools for neuroscientific research and technical applications [4, 31, 32, 33]. All analog circuits, however, suffer from device variations caused by unavoidable variability in the manufacturing process. Neurons and synapses implemented in analog neuromorphic hardware therefore exhibit heterogeneous response properties, similar to their biological counterparts [34, 35]. To understand the dynamics and function of recurrent neural networks in both biological and artificial substrates, it is therefore essential to account for such heterogeneities.

Previous studies on recurrent neural networks have shown that heterogeneity in single-neuron properties or connectivity broadens the distribution of firing rates [36, 20] and affects the stability of asynchronous or oscillatory states [37, 38, 39, 40, 27, 41]. A number of studies pointed at a potential benefit of heterogeneity for the information-processing capabilities of neural networks [42, 43, 44, 41, 45, 46]. The effect of heterogeneity on correlations in the activity of recurrent networks of spiking neurons, however, remains unclear yet. Roxin [27] pointed out that heterogeneity in the number of outgoing connections (out-degree) increases the fraction of shared input and may therefore lead to an increase in correlations. Padmanabhan & Urban [43] have shown that the responses of a population of unconnected neurons are decorrelated by heterogeneity in the neuronal response properties. These results are supported by the subsequent theoretical analysis in [44]. In the following, we refer to this type of decorrelation by heterogeneity as *feedforward decor-*

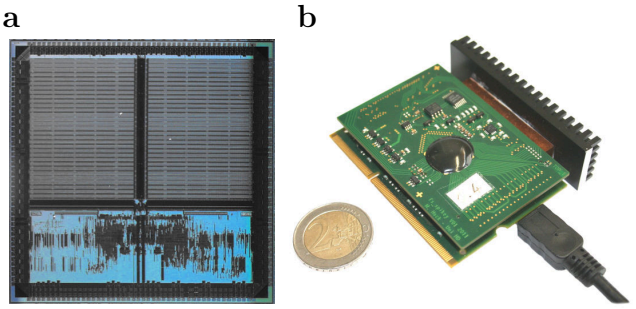


Figure 1: The neuromorphic hardware system *Spikey*. **(a)** Photograph of the *Spikey* chip (size $5 \times 5\,\mathrm{mm}^2$). It comprises analog circuits of 384 neurons and 98304 synapses (half of which are accessible for chip version 4 used in this study), is highly configurable and emulates neural-network dynamics by a factor 10^4 faster than biological real-time. **(b)** Photograph of the *Spikey* system, carrying the *Spikey* chip (covered by a black round seal) and conventional memory. The system is connected to the host computer via USB 2.0, consumes 6 W of power in total and less than 1 nJ per synaptic transmission (see Supplements 1).

relation. It does not account for the effect of the recurrent network dynamics. Active decorrelation due to inhibitory feedback [see above; 13, 14], in contrast, constitutes a very different mechanism. The effect of heterogeneity on this *feedback decorrelation* has lately been studied by Bernacchia & Wang [47] in the framework of a recurrent network of linear firing-rate neurons. In this setup, correlations are suppressed by heterogeneity in the network connectivity (distributions of coupling strengths or random dilution of connectivity). It remains unclear, however, whether this holds true for networks of (nonlinear) spiking neurons.

In this study, we investigate the impact of heterogeneity on input and output correlations in sparse networks of leaky integrate-and-fire (LIF) neurons with conductance-based synapses, implemented in the analog neuromorphic hardware system *Spikey* (Figure 1) [48, 49]. The configurability of this system [49] enables us to systematically vary the level of heterogeneity, and to disentangle the effects of heterogeneity on feedforward and feedback decorrelation (see above). For simplicity, we focus on purely inhibitory networks, thereby emphasizing that active decorrelation by inhibitory feedback does not rely on a dynamical balance between excitation and inhibition [14, 50]. We show that decorrelation by inhibitory feedback is effective even in highly heterogeneous networks with broad distributions of firing rates (Section 3.1). Increasing the level of heterogeneity has two effects: Feedforward decorrelation is enhanced, feedback decorrelation is impaired. Overall, input and output correlations are increased (Section 3.2). Note that the findings presented in this article are acquired by network emulations on analog neuromorphic hardware. Qualitatively similar results are however obtained by means of simulations using conventional computers (see Supplements 3).

2 Methods

2.1 Network model

Details on the network, neuron and synapse model are provided in Table 1. Parameter values are given in Table 2. Briefly: We consider a purely inhibitory, sparse network

of N ($N = 192$, unless stated otherwise) LIF neurons with conductance-based synapses. Each neuron receives input from a fixed number $K = 15$ of randomly chosen presynaptic sources, independently of the network size N. Self-connections and multiple connections between neurons are excluded. Resting potentials E_l are set above the firing thresholds Θ (equivalent to applying a constant supra-threshold input current). We thereby ensure autonomous firing in the absence of any further external input. Due to temporal noise, the initial conditions are essentially random.

2.2 Network emulations on the neuromorphic hardware system *Spikey*

The *Spikey* chip consists of physical models of LIF neurons and conductance-based synapses with exponentially decaying dynamics (Figure 1; for details, see Table 1). The emergent dynamics of these physical models represents a solution for the model equations of neurons and synapses in continuous time, and in parallel for all units. In contrast, in numerical simulations model equations are solved by stepwise integration, where parallelization is limited by the available number of virtual processes. To emphasize the difference between simulations using software and simulations using physical models the term *emulation* is used for the latter [49].

The response properties of physical neurons and synapses vary across the chip due to unavoidable variations in the production process that manifest in a spatially disordered pattern (*fixed-pattern noise*). In contrast to the approximately static fixed-pattern noise, *temporal noise*, including electronic noise and transient experiment conditions (e.g., chip temperature), impairs the reproducibility of emulations. Two network emulations with identical configuration and stimulation do generally not result in identical network activity. Both fixed-pattern and temporal noise need to be taken into account when developing models for analog neuromorphic hardware.

The key features of the *Spikey* chip are the high acceleration and configurability of the analog network implementation. Some network parameters, e.g., synaptic weights and leak conductances, are configurable for each unit, while other parameters are shared for several units (for details see [49]). The hardware system is optimized for spike in- and output and allows to record the membrane potential of one (arbitrarily chosen) neuron with a sampling frequency of 96 MHz in hardware time. Networks on the *Spikey* chip are emulated much faster (approximately 10^4-fold) than biological real-time, which is a direct consequence of the small capacitances and much higher conductances of VLSI technology compared to biological nervous systems. Due to this high acceleration of the neuromorphic chip, the data bandwidth of the connection between the neuromorphic system and the host computer is not sufficient to communicate with the chip in real time. Consequently, input and output spikes (for stimulation and from recordings, respectively) are buffered in a local memory next to the chip. The high acceleration of the *Spikey* chip allows to operate most of the transistors outside of weak inversion, thereby reducing the effect of transistor variations and minimizing fixed-pattern noise.

In contrast to such accelerated systems, most other analog neuromorphic substrates are designed for real-time emulations at very low power consumption [51, 52, 53, 54, 55, 56, 57] and implement a few but complex neurons [58, 59].

Access to the *Spikey* system is encapsulated by the simulator-independent language `PyNN` [60, 61], providing a stable and user-friendly interface. `PyNN` integrates the hardware into the computational neuroscience tool chain and has facilitated the implementation of several network models on the *Spikey* chip [62, 63, 64, 49, 4].

On the *Spikey* system, a spiking neural network is emulated as follows (Figure 2a): First, the network described in `PyNN` is mapped to the *Spikey* chip, i.e., neurons and

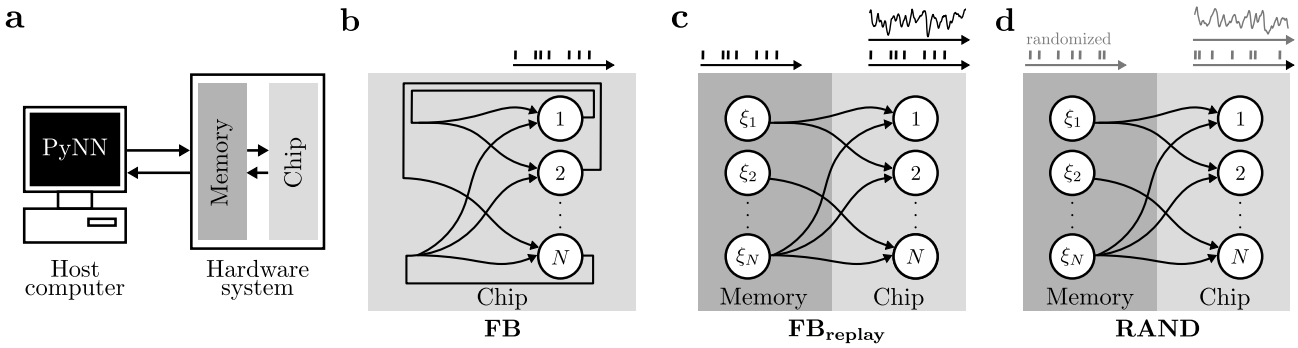


Figure 2: Experimental setup. **(a)** Data flow of the *Spikey* system. For details see Section 2.2. **(b)** Network with on-chip feedback connections (FB). Spikes from all neurons are recorded to the local memory. **(c)** Spikes of the FB network in (b) replayed from memory via off-chip spike sources ξ_i to neurons i (FB$_{\text{replay}}$). Spike times of ξ_i correspond to those recorded from neuron i in (b). Spikes from all neurons or the free membrane potential of one selected neuron are recorded. **(d)** Like (c), but spike times from (b) are randomized for each source (RAND).

synapses are allocated and parametrized. Second, input spikes, if available, are prepared on the host computer and transferred to the local memory on the hardware system. Third, the emulation is triggered and available input spikes are generated. Output spikes and membrane data are recorded to local memory. Last, spike and membrane data are transferred to the host computer and scaled back into the biological domain of the `PyNN` model description.

For consistency with the model description and simplified comparison to the existing literature, all hardware times and all hardware voltages are expressed in terms of the quantities they represent in the neurobiological model, throughout this study.

2.3 Experimental setup

To differentiate and compare the effects of shared inputs and feedback connections on correlations, we investigate two different emulation scenarios: First, we emulate networks with intact feedback (FB, Figure 2b), and second, the contribution of shared input is isolated by *randomizing* the temporal order of this feedback (RAND, Figure 2d).

In the RAND scenario, the inputs of neurons are decoupled from their outputs. The activity of the previously recorded FB network is *replayed* to a population of unconnected neurons of equal size. We keep the connectivity the same, but randomize the presynaptic spike times. Each neuron hence receives the same number of spikes as in the recurrent network during the whole emulation, but spatio-temporal correlations in presynaptic spike trains are removed. To preserve the fixed pattern of variability of synaptic weights in hardware, the same hardware synapses are used for each connection in both scenarios.

Input correlations between neurons are measured via their *free membrane potential*, i.e., the membrane potential with disabled spiking mechanism (technically, the threshold is set very high). Because membrane potential traces can be recorded in the hardware only one at a time, traces are obtained consecutively while repeatedly replaying the activity of the FB network without randomization (FB$_{\text{replay}}$, Figure 2c) or replaying randomized activity (RAND), respectively. If network dynamics were reproduced perfectly, membrane potential traces and spike times would be identical in the FB and FB$_{\text{replay}}$ cases (see also Section 2.4).

Drawing two different network realizations (i.e., the connectivity matrix) results in the allocation of different hardware synapses, and, due to fixed-pattern noise, in different values of synaptic weights. To average over this variability, throughout this study, emulation results are averaged over $M = 100$ network realizations, if not stated otherwise.

2.4 Reproducibility of hardware emulations

In contrast to the chaotic activity in the FB network (see however [65]), in the RAND and $\text{FB}_{\text{replay}}$ case the input of neurons is decoupled from their output. Therefore, even in the presence of the unavoidable temporal noise, the system's trajectory tends to return to the trajectory of the noiseless case. A certain degree of reproducibility is required for two reasons: First, the investigated effect of decorrelation by inhibitory feedback requires a precise relation between spike input and output. Thus our method of replacing the feedback loop by replay is only valid if temporal noise does not substantially corrupt this relationship. Second, to record the membrane potentials of all neurons, as if recorded at once, neuron dynamics have to be reasonably similar in consecutive emulations.

We measure the reproducibility of neuron dynamics by comparing consecutive emulations with identical configuration, i.e., connectivity and stimulation. For this purpose the spiking activity of a FB network is first recorded (Figure 2b) and then repeatedly replayed (Figure 2c). Reproducibility is quantified by the correlations (κ_X in Table 3) of free membrane potential traces and output spike trains obtained for individual neurons in $L = 25$ different trials.

Free membrane potentials are reproduced quite well, while spike trains show larger deviations across trials (Figure 3). Small deviations in the membrane potential (Figure 3b) are amplified by the thresholding procedure [66, 67, 68] and can lead to large differences between spike trains (Figure 3c). Consequently, measures based on data of several consecutive replays are more precise for membrane potentials than for spike trains. Nevertheless, results have to be interpreted with care in both cases.

2.5 Calibration

The heterogeneity of the *Spikey* hardware is adjusted by calibrating the leak conductance[1] for each individual neuron, compensating for fixed-pattern noise of neuron parameters. To this end, a population of unconnected neurons is driven by a suprathreshold constant current influx and the time-averaged population activity $\bar{r}$ is measured. Then, we applied the bisection method [69] to adjust the leak conductance g_{l} of each neuron, such that the neuron's firing rate matches the target rate $\bar{r}$. This results in calibration values b for the leak conductance $g_{\text{l}} = g_{\text{l},0}(1 + b)$, where $g_{\text{l},0}$ is the leak conductance before calibration. Because emulations on hardware are not perfectly reproducible, more precise calibration was achieved by evaluating the median over 25 identically configured trials instead of single trials. Furthermore, the bisection method was modified for noisy systems (for details, see Supplements 2).

Intermediate calibration states are obtained by linearly scaling the full calibration:

$$g_{\text{l}} = g_{\text{l},0}(1 + (1 - a)b) \quad . \tag{1}$$

The heterogeneity a is chosen in $[0, 1]$ for calibrations between the uncalibrated ($a = 1$) and calibrated state ($a = 0$). In the following, the fully calibrated chip ($a = 0$) is used, if not stated otherwise.

[1] since capacitances and potentials can not be configured individually for each hardware neuron [49]

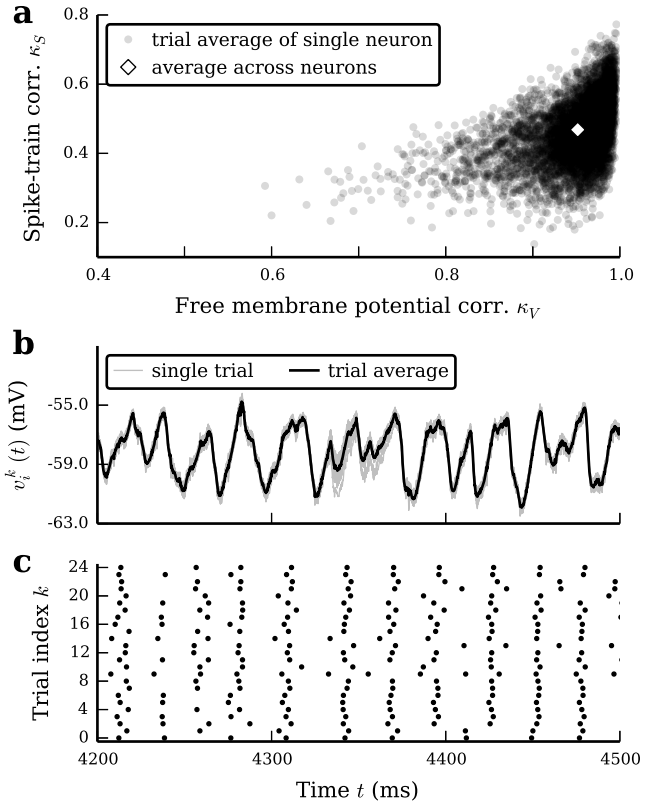


Figure 3: Reproducibility of free membrane potentials and spiking activity in the $\mathrm{FB}_{\mathrm{replay}}$ case. **(a)** Low-frequency coherence κ_V and κ_S of free membrane potentials $v_{\mathrm{i}}^k(t)$ and $v_{\mathrm{i}}^l(t)$ and binned spike trains $s_{\mathrm{i}}^k(t)$ and $s_{\mathrm{i}}^l(t)$, respectively, for each neuron i averaged over $L = 25$ trials k, l with $k \neq l$, for $M = 50$ different network realizations. The diamond marks the average across all neurons i and M network realizations. **(b)** Free single-trial membrane potentials $v_{\mathrm{i}}^k(t)$ (gray) and average over trials $\frac{1}{L}\sum_{k=1}^{L} v_{\mathrm{i}}^k(t)$ (black) and **(c)** spike density $\xi_{\mathrm{i}}(t)$ of a single neuron i for $L = 25$ identical trials. The selected neuron i has membrane potential coherence and spike train coherence closest to the diamond in (a).

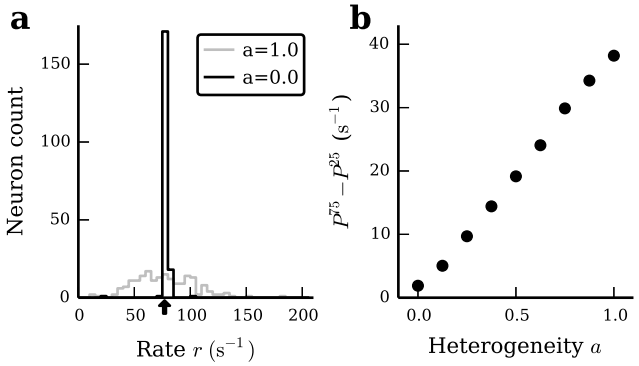


Figure 4: Calibration of the *Spikey* chip. **(a)** Histogram of firing rates r for a population of unconnected neurons before (gray) and after (black) calibration, each neuron averaged over $L = 100$ trials. The arrow denotes the target rate $\bar{r}$. **(b)** Difference $P^{75} - P^{25}$ of 75th and 25th percentile of histograms in (a) as a function of network heterogeneity a (Equation 1). The mean firing rate over all values of a is $(78.1 \pm 0.7)\,\mathrm{s}^{-1}$.

This calibration substantially narrows the distribution of firing rates compared to the uncalibrated state (Figure 4). With respect to the stationary firing rate, variability on the neuron level is reduced from 38% to 2%.

Note that after this calibration procedure the hardware network is still not homogeneous. In addition to remaining variations in neuron parameters, synaptic parameters have a significant variation [70, 4].

2.6 Correlation measures

In the following, we introduce definitions used to analyze the recorded data. For clarity, all relevant equations and their parametrization are listed in Table 3 and 4, respectively.

We quantify correlations of membrane potentials $v_\mathrm{i}(t)$ and spike trains $s_\mathrm{i}(t)$ by the population-averaged *low-frequency coherence* κ_V and κ_S, respectively. At frequency zero, the coherence corresponds to the normalized integral of the cross-covariance function, i.e., it measures correlations on all time scales. We define the low-frequency coherence κ_X, with $X \in \{S, V\}$, to be the average coherence over a frequency interval from 0.1 to 20 Hz. In this interval, the suppression of population-rate fluctuations in recurrent networks due to inhibitory feedback is most pronounced, and the coherence is approximately constant. Before calculating the coherence, we average the power- and cross-spectra with a sliding window to average out random fluctuations. This measure, or a variant of it, is commonly used in the neuroscientific literature [71, 72, 73, 67, 13, 14, 44]. We use the terms low-frequency coherence and correlation interchangeably.

Throughout this study, the term *input correlations* is used for correlations between free membrane potentials, and *output correlations* for correlations between spike trains. *Shared-input correlations* are membrane potential correlations that are exclusively caused by overlapping presynaptic sources, ignoring possible correlations in the presynaptic activity. The average pairwise shared-input correlations in a homogeneous network are of the size of the connectivity [14]:

$$\kappa_V = K/N \; . \tag{2}$$

To quantify fluctuations in the population activity $\bar{s}$ (Figure 5a–c, horizontal histograms) we compute the power spectrum $\bar{A}$ of the population activity (Figure 5e), which

we scale with the duration T of the emulation. Consequently, the population power spectrum $\bar{A}(f)$, scaled by the population size, coincides with the time-averaged population activity $\bar{r}$ for high frequencies: $\lim_{f\to\infty} \frac{1}{N}\bar{A}(f) = \bar{r}$ [11].

As a measure of pairwise correlations in the time domain (Figure 5d), we compute the population-averaged cross-correlation function $c(\tau)$ by Fourier transforming the population-averaged cross-spectrum $C(f)$ to time domain.

3 Results

In this study, we investigate the roles of shared input, feedback and heterogeneity on input and output correlations in random, sparse networks of inhibitory LIF neurons with conductance-based synapses (Table 1), implemented on the analog neuromorphic hardware chip *Spikey* (Figure 1). Similarly to [14], we separate the contributions of shared input and feedback by studying different network scenarios (Figure 2): In the FB case, we emulate the recurrent network with intact feedback loop (Figure 2b) and record its spiking activity (Figure 5a). In the $\text{FB}_{\text{replay}}$ case (Figure 2c), the feedback loop is cut and replaced by the activity recorded in the FB network. Ideally, the input to each neuron in the $\text{FB}_{\text{replay}}$ case should be identical to the input of the corresponding neuron in the FB network. As the replay of spikes and the resulting postsynaptic currents and membrane potentials are not perfectly reproducible on the *Spikey* chip, the neural responses in the FB and in the $\text{FB}_{\text{replay}}$ scenario are slightly different (compare Figure 5 a to b). In the RAND case (Figures 2d and 5c), we use the same setup as in the $\text{FB}_{\text{replay}}$ case. However, the spike times in each presynaptic spike train are randomized. While the average presynaptic firing rates and the shared-input structure are exactly preserved in this scenario, the spatio-temporal correlations in the presynaptic spiking activity are destroyed.

Using this setup, we first demonstrate in Section 3.1 that active decorrelation by inhibitory feedback [13, 14] is effective in heterogeneous networks with conductance-base synapses over a range of different network sizes. In Section 3.2, we show that decreasing the level of heterogeneity by calibration of hardware neurons leads to an enhancement of this active decorrelation and thereby to a decrease in input and output correlations.

3.1 Decorrelation by inhibitory feedback

The time-averaged population activities in the FB, $\text{FB}_{\text{replay}}$ and RAND scenarios are roughly identical (Figure 5a–c; see also high-frequency power in Figure 5e). In the FB and $\text{FB}_{\text{replay}}$ scenario, fluctuations in the population-averaged activity are small (horizontal histograms in Figure 5a and b). The removal of spatial and temporal correlations in the presynaptic spike trains in the RAND case leads to a significant increase in the fluctuations of the population-averaged response activity (horizontal histograms in Figure 5c). At low frequencies ($\leq 20\,\text{Hz}$), the population-rate power in the FB and in the RAND case differs by almost two orders of magnitudes (black dotted and gray curves in Figure 5e). This increase in low-frequency fluctuations in the RAND case is mainly caused by an increase in pairwise correlations in the spiking activity (Figure 5d; the power spectra of individual spike trains [inset in Figure 5e] are only marginally affected by a randomization of presynaptic spike times) [14]. In other words, shared-input correlations, i.e., those leading to large spike-train correlations in the RAND scenario, are efficiently suppressed by the feedback loop in the FB case.

On the neuromorphic hardware, the replay of network activity is not perfectly reproducible (Section 2.4). While the across-trial variability in membrane potentials is small,

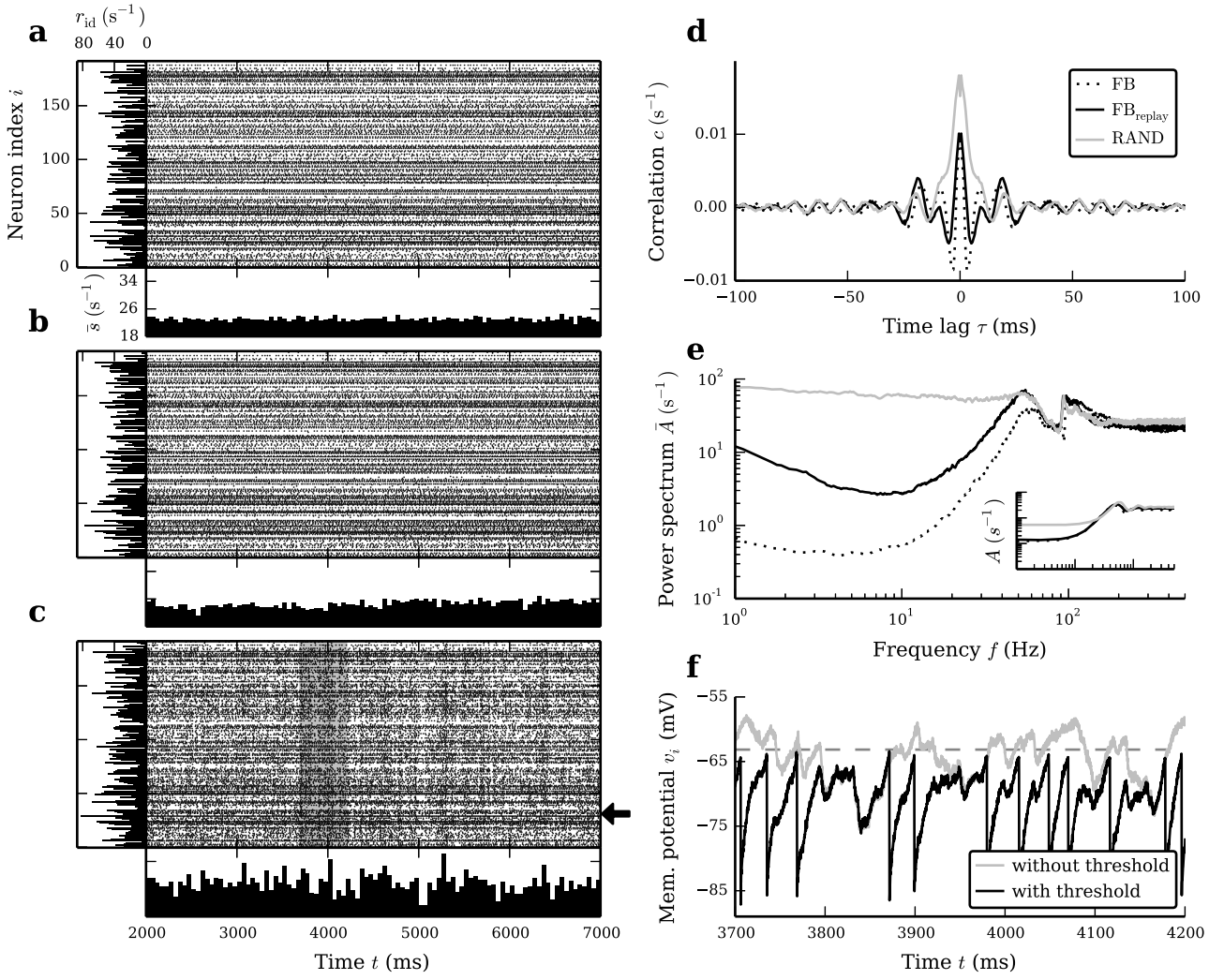


Figure 5: Spiking and membrane-potential activity in a random inhibitory network of LIF neurons with intact and cut feedback loop. **(a–c)** Spiking activity (raster plots), population activity $\bar{s}(t)$ (horizontal histograms; bin size 50 ms) and time-averaged single-neuron firing rates r_{id} (vertical histograms) in the network with intact feedback (a) and for cases where the feedback loop is cut (b and c). (a) Intact recurrent network (FB scenario). (b) Population of mutually unconnected neurons receiving identical input spike trains as in (a) (FB$_{\mathrm{replay}}$ scenario). (c) As in (b), but after randomization of presynaptic spike times (RAND scenario). **(d and e)** Population-averaged cross-correlation functions $c(\tau)$ (after offset subtraction) of pairs of spike trains (d) and power spectra $\bar{A}(f)$ (e; log-log representation) of the population activity $\bar{s}(t)$ (cf. horizontal histograms in (a–c)) for the FB (dotted), FB$_{\mathrm{replay}}$ (solid black) and RAND scenario (gray). Inset in (e): Population-averaged power spectra $A(f)$ of individual single-cell spike trains (same scales as in main panel). Correlation functions and spectra are averaged across $M = 100$ network realizations. **(f)** Membrane potential of a neuron in the RAND scenario (with firing rate of 29.0 s^{-1} close to population average of 27.8 s^{-1}; see black arrow in (c)) with intact (black curve) and removed threshold (gray curve; *free membrane potential*). The threshold potential is marked by the horizontal dashed line. The time frame corresponds to the gray-shaded region in (c).

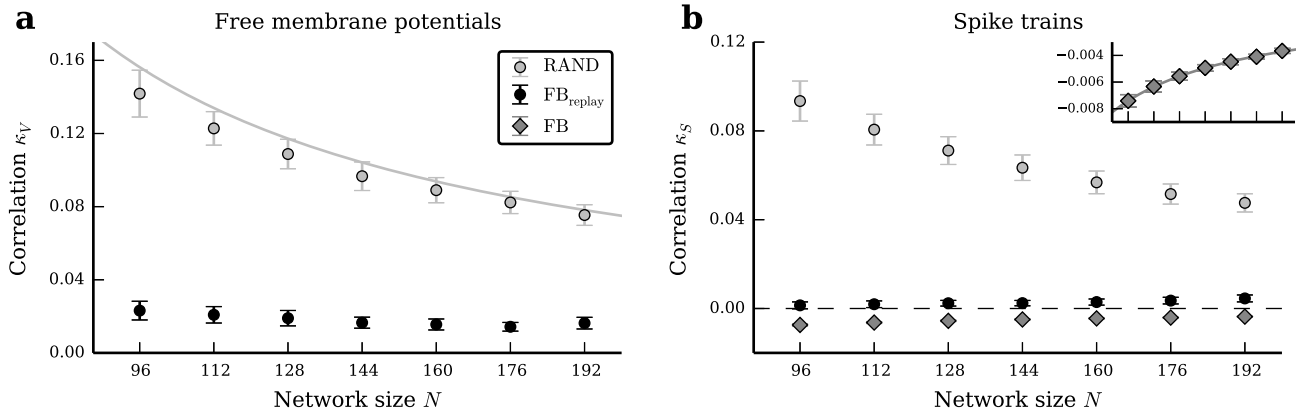


Figure 6: Dependence of population-averaged input correlations **(a)** and spike-train correlations **(b)** on the network size N for the intact network (FB, dark gray), the $\text{FB}_{\text{replay}}$ (black) and the RAND (light gray) case (fixed in-degree $K = 15$). Symbols and error bars denote mean and standard deviation, respectively, across $M = 100$ network realizations (errorbars are partly covered by markers). Gray curve in (a) depicts shared-input correlations in a homogeneous network (Equation 2). Black dashed line in (b) marks zero for orientation. The inset in (b) shows a magnified view of the spike-train correlations in the FB case (diamonds) with a power-law fit $\sim N^{-1}$ (dark gray curve). Note that free membrane potentials cannot be recorded in the FB case (see Section 2). Hence, there are no gray diamonds in (a).

postsynaptic spikes are jittered on a timescale of approximately 5 ms (Figure 3). In the $\text{FB}_{\text{replay}}$ case, the suppression of shared-input correlations by correlations in presynaptic spike trains is therefore slightly less efficient as compared to the intact network (FB). The differences in the population-rate power spectra and in the spike-train correlations between the $\text{FB}_{\text{replay}}$ and RAND case, respectively, are nevertheless substantial (solid black and gray curves in Figure 5d and e).

In the RAND case, input (i.e., free-membrane-potential) correlations are exclusively determined by the number of shared presynaptic sources, i.e., by the connectivity K/N, and, hence, decrease with network size N if the in-degree K is fixed (gray curve and symbols in Figure 6; see also Equation 2 and [14, 74]). In the FB scenario, two components contribute to the input correlation: Shared input and correlations in presynaptic spike trains. In purely inhibitory networks, the average spike-train correlation is negative (diamonds in Figure 6b) [14]. Shared-input correlations, which are always positive if Dale's law is respected [10], are largely canceled by these negative spike-train correlations. The average input correlations are therefore significantly reduced (black symbols in Figure 6a). As both spike-train and shared-input correlations scale with the inverse N^{-1} of the network size (Figure 6a and inset in Figure 6b) [67], this suppression of correlations in the FB (and $\text{FB}_{\text{replay}}$) case is observed for all investigated network sizes N. Note that the output correlations are negative even though input correlations are positive. This effect is predicted by theory and also observed in linear network models as well as LIF-network simulations on conventional computers (see Supplements 3, Supplements 4 and Section 4).

3.2 Effect of heterogeneity on decorrelation

In neural networks implemented in analog neuromorphic hardware, neuron (and synapse) parameters vary significantly across the population of cells (fixed-pattern noise; see Section 2.2). For a population of mutually unconnected neurons with distributed parameters, injection of a constant (suprathreshold) input current leads to a diversity of response firing rates (Figure 4). In this study, we consider the width of this firing-rate distribution as a representation of neuron heterogeneity. It is systematically varied by partial calibration of leak conductances. The extent of heterogeneity is quantified by the calibration parameter a ($a = 1$ and $a = 0$ correspond to the uncalibrated and the fully calibrated system, respectively; for details, see Section 2.5). For an unconnected population of neurons subject to constant input, the width of the firing-rate distribution increases monotonically with a.

As shown in Figure 7, the level of heterogeneity (i.e., the calibration state a) is clearly reflected in the activity of the wired recurrent network (FB case). Both the width of the distribution of mean free membrane potentials (Figure 7a–c) as well as the width of the firing-rate distribution increase with a (Figure 7d–f; bottom panels). In the uncalibrated system ($a = 1$), a substantial fraction of neurons is predominantly driven by constant suprathreshold input currents and therefore generates highly regular spike trains ($C_{\mathrm{V}}^{ISI} \approx 0$) with high firing rates ($> 120\,\mathrm{s}^{-1}$). Simultaneously, about 40% of the neurons are silent ($0\,\mathrm{s}^{-1}$). Both highly active and inactive neurons are hardly modulated by (inhibitory) recurrent inputs from the local network. After calibration, the firing-rate distribution is narrowed. For $a = 0$, the fraction of silent neurons is reduced to about 10%. Maximum rates are limited to $< 80\,\mathrm{s}^{-1}$. Note that our calibration routine compensates only for the distribution of neuron parameters, but not for the heterogeneity in synapse properties (synaptic weights, synaptic time constants; see Section 4). For the fully calibrated network ($a = 0$), the firing-rate distribution is therefore still broad. In the RAND case, we obtain similar firing-rate and inter-spike interval statistics as in the FB case (Supplements 5).

For all levels of heterogeneity attainable by our calibration procedure ($a \in [0, 1]$), input and output correlations are significantly suppressed by the recurrent-network dynamics (cf. black and dark gray vs. light gray symbols in Figure 8). In a homogeneous, random (Erdős-Rényi) network with fixed in-degree K and linear subthreshold dynamics, the contribution of shared input to the input (free-membrane-potential) correlation is given by the network connectivity K/N [14] (gray curves in Figure 6a and Figure 8a). Nonlinearities in synaptic and/or spike-generation dynamics [67] as well as heterogeneity in neuron (and synapse) parameters lead to a suppression of this contribution [43]. Here, we refer to this type of decorrelation as *feedforward decorrelation*. In fact, in our setup the spike-train correlations in the RAND case slightly decrease with increasing heterogeneity (light gray symbols in Figure 8b). The input correlations in the RAND case, in contrast, are marginally affected by the calibration and only slightly smaller than K/N (gray symbols vs. gray curve in Figure 8a). This observation may indicate that the dominant source of heterogeneity in our networks results from distributions of parameters which affect the spike-generation (spike thresholds Θ, leak conductances g_{l}, resting potentials E_{l}) or after-spike dynamics (reset potentials v_{reset}, refractory periods τ_{ref}), but not the integration of synaptic inputs. Broad distributions of synaptic weights J, inhibitory reversal potentials E_{inh}, membrane or synaptic time constants τ_{m}, τ_{syn} or delays d would lead to a feedforward decorrelation also at the level of the free membrane potential. We mimicked the effect of threshold heterogeneity in network simulations on conventional computers and obtain results which are qualitatively similar to those shown here (Supplements 3).

Although feedforward decorrelation benefits from cell heterogeneity, input and output correlations grow with the level of heterogeneity in the presence of an intact feedback

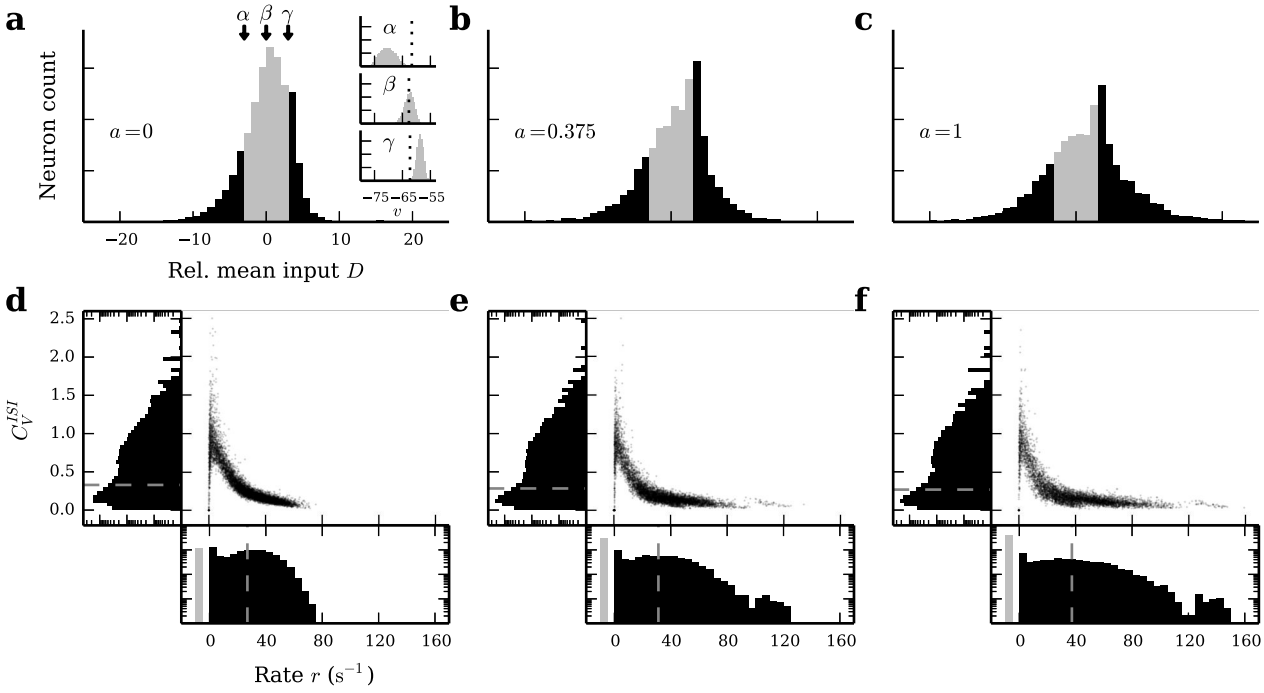


Figure 7: Modulation of network heterogeneity by leak-conductance calibration (see Section 2.5). Input (top row) and firing statistics (bottom row) in the intact recurrent networks (FB scenarios) for fully calibrated (a and d; $a = 0$), partially calibrated (b and e; $a = 0.625$) and uncalibrated neurons (c and f; $a = 1$). **(a–c)** Effect of calibration on input statistics. Distributions of relative mean input $D = (\bar{v} - \Theta)/\sigma(v)$ (distance of time averaged free membrane potential $\bar{v}$ from firing threshold Θ in units of the standard deviation $\sigma(v)$) across the population of neurons. Gray areas in (a), (b) and (c) highlight $[-3, 3]$ intervals, containing 88%, 69% and 60% of the total mass of the distribution, respectively. Inset in (a): Distributions of free membrane potentials v for three neurons α, β and γ with $D = -3$, $D = 0$ and $D = 3$ (arrows in (a)), respectively. Dotted lines mark threshold potentials that may vary due to fixed-pattern noise. **(d–f)** Effect of calibration on spike-train statistics. Joint (scatter plots) and marginal distributions of single-neuron firing rates r (horizontal histograms; log-linear scale) and coefficients of variation C_V^{ISI} of inter-spike intervals (vertical histograms; log-linear scale). Dashed lines mark mean of firing rate ($26.9\,\mathrm{s}^{-1}$, $31.2\,\mathrm{s}^{-1}$, $37.3\,\mathrm{s}^{-1}$) and C_V^{ISI} distributions (0.33, 0.28, 0.27), respectively. Gray bars (bottom panels) represent fractions of silent neurons. Data obtained from $M = 50$ different network realizations.

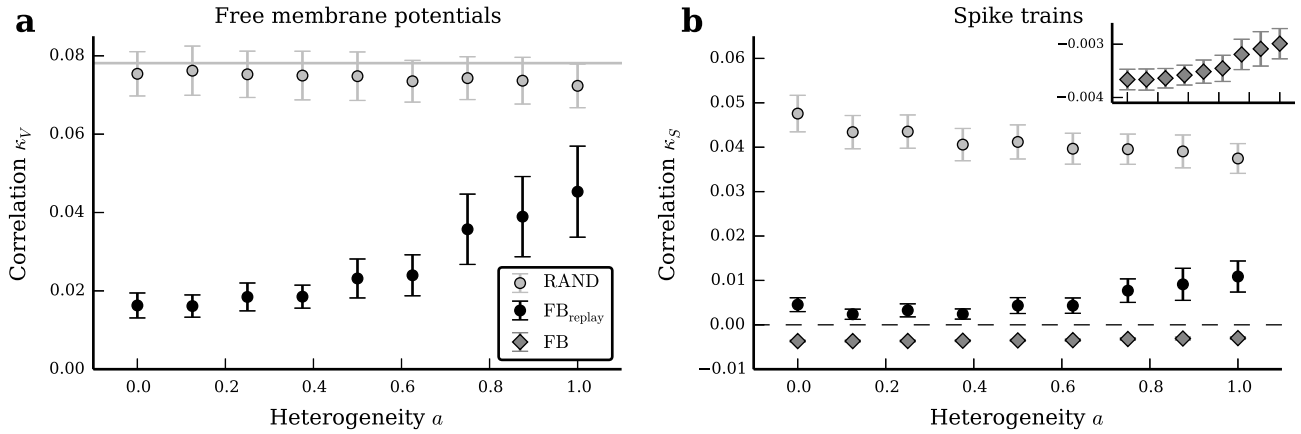


Figure 8: Dependence of population-averaged input correlations **(a)** and spike-train correlations **(b)** on the heterogeneity of the neuromorphic substrate for the intact network (FB, dark gray), the $\text{FB}_{\text{replay}}$ (black) and the RAND (light gray) case. Symbols and error bars denote mean and standard deviation, respectively, across $M = 100$ network realizations (errorbars are partly covered by markers). Gray curve in (a) depicts shared-input correlations in a homogeneous network (Equation 2). Black dashed line in (b) marks zero for orientation. The inset in (b) shows a magnified view of the spike-train correlations in the FB case (diamonds). Note that free membrane potentials cannot be recorded in the FB case (see Section 2). Hence, there are no gray diamonds in (a).

signal (black and dark gray symbols in Figure 8). We attribute this effect to a weakening of the effective feedback loop in the recurrent circuit: In heterogeneous networks with broad firing-rate distributions, neurons firing with very low or high rates (corresponding to mean inputs far below or far above firing threshold; see Figure 7a–c) are less sensitive to input fluctuations than moderately active neurons. Hence, they contribute less to the overall feedback. In consequence, feedback decorrelation is impaired (see also Section 4).

4 Discussion

We have shown that inhibitory feedback effectively suppresses correlations in heterogeneous recurrent neural networks of leaky integrate-and-fire (LIF) neurons with nonlinear sub-threshold dynamics , emulated on analog neuromorphic hardware (*Spikey*; [48, 49]). Both input and output correlations are substantially smaller in networks with intact feedback loop (FB) as compared to the case where the feedback is replaced by randomized input (RAND). The study hence confirms that active decorrelation of network activity by inhibitory feedback [13, 14] is a general phenomenon which can be observed in realistic, highly inhomogeneous networks with sufficiently strong negative feedback.

Partial calibration of hardware neurons allowed us to modulate the level of network heterogeneity and, therefore, to systematically study its effect on correlations in the network activity. The analysis revealed two counteracting contributions: As shown in previous studies [e.g. 43], neuron heterogeneity decorrelates (shared) feedforward input (feedforward decorrelation). On the other hand, however, heterogeneity impairs feedback decorrelation (see next paragraph). In our network model, this weakening of feedback decorrelation is the dominating factor. Overall, we observed an increase in correlations with increasing level of heterogeneity. We cannot exclude that feedforward decorrelation may play a more

significant role for different network configurations (e.g., different connection strengths or network topologies, different structure of external inputs, other types of heterogeneity). Our study demonstrates, however, that heterogeneity is not necessarily suppressing correlations in recurrent systems.

As shown in [14], feedback decorrelation in recurrent networks becomes more (less) efficient with increasing (decreasing) strength of the *effective* negative feedback. For networks of spiking neurons, the effective connection strength w_{ij} (also termed DC susceptibility [67]) between two neurons j and i corresponds to the total number of extra spikes emitted by neuron i in response to an additional input spike (perturbation) generated by neuron j. Assuming that the effect of a single additional input spike is small, the effective connectivity can be obtained by linear-response theory. Note that the effective weights w_{ij} depend on the working point, i.e., the average firing rates of all pre- and postsynaptic neurons (mathematically, w_{ij} is given by the derivative of the stationary response firing rate $r_i = \phi_\mathrm{i}(r_1, \ldots, r_j, \ldots, r_N)$ with respect to the input firing rate r_j, evaluated at the working point; for details, see [14]). Neurons firing at very low or very high rates are typically less sensitive to input fluctuations than neurons firing at intermediate rates (due to the shape of the response function $\phi_\mathrm{i}(r_1, \ldots, r_N)$). Their dynamical range is reduced. In consequence, they can hardly mediate the feedback in a recurrent network. In heterogeneous networks with broad distributions of firing rates, the number of these insensitive neurons is increased. Hence, the effective feedback is weakened. In Supplements 4, we mimic the effect of heterogeneity by decreasing the effective weights in a linear rate model. The resulting dependence of input and output correlations on the level of heterogeneity qualitatively resembles those results we obtained for the nonlinear spiking network emulated on the neuromorphic system. A direct quantitative comparison between both models requires an explicit mapping of the synaptic weights in the LIF-neuron network to the effective weights of the linear model in the presence of distributed firing rates. We commit this task to future studies. Note that the rate dependence of the effective weights and the resulting effects of heterogeneity are consistent with our observation that LIF-neuron pairs with very low firing rates exhibit spike-train correlations close to zero, whereas pairs with high firing rates are positively correlated. Pairs with at least one neuron firing at an intermediate rate (the second neuron can fire at a higher rate) exhibit negative spike-train correlations (see Supplements 6). As shown in [13, 14], these negative spike-train correlations are essential for compensating the positive contribution of shared inputs to the total input correlation (at least in purely inhibitory networks). Narrowing the firing rate distribution (e.g., by calibration of hardware neurons) increases the number of neurons contributing to the negative feedback, which, in turn, leads to more neuron pairs with negative spike-train correlations and, therefore, to smaller overall correlations.

Seemingly contrary to our findings, Bernacchia & Wang [47] report a decrease in correlations with increasing level of heterogeneity. The results of their study are obtained for a linear network model, which can be considered the outcome of the linearization procedure described above. Hence, the connectivity of their model corresponds to an effective connectivity (see above). Their study neglects the rate (working-point) dependence of the effective weights and can therefore not account for the effect of firing-rate heterogeneity. In [47], heterogeneity is quantified by the variance of the (effective) weight matrix (Equations 2.2 and 2.4 in [47]). For sparse connectivity matrices (with a large number of zero elements), the variance of the weight matrix reflects not only the width of the non-zero-weight distribution, but also its mean (Equation 2.4 in [47]). For networks of nonlinear spiking neurons, heterogeneities in neuron and/or synapse parameters broadens the distribution of non-zero effective weights, but may simultaneously reduce its mean (see above

and [20, 75]). Hence, the variance of the full weight matrix may decrease (for illustration, see Supplements Figure 3). In other words, increasing heterogeneity in the nonlinear system may correspond to decreasing heterogeneity in the linearized system. A direct test of this hypothesis requires an explicit linearization of the nonlinear heterogeneous system, which, again, may be subject of future studies.

The results of this study were obtained by network emulations on analog neuromorphic hardware. We reproduced the main findings by means of simulations of LIF-neuron networks with distributed firing thresholds on conventional computers (see Supplements 3). Although networks simulated on conventional computers and those emulated on the neuromorphic hardware differ in several respects (e.g., in the exact implementation of heterogeneity or the synapse model; see Supplements Table 1 and 2), the qualitative results are very similar: In networks with intact feedback loop, input and output correlations are substantially reduced (as compared to the case where the feedback is replaced by randomized input), but increase with the extent of heterogeneity. As predicted by the theory for homogeneous inhibitory networks, we observe positive input correlations and negative output correlations (see Equation 21 in [14] and paragraph thereafter; see also [74] and Supplements 4). Further, note that heterogeneity in neuron parameters does not "average out" in larger networks. Upscaling the network size by a factor of 25 ($N = 4800$, in-degree $K = 384$) yields smaller spike-train correlations, but the qualitative results are similar to those obtained for the smaller network ($N = 192$, $K = 15$) emulated on the *Spikey* chip (compare Figure 8 to Supplements Figure 1).

In networks with intact feedback loop (FB scenarios), the precise spatio-temporal structure of spike trains arranges such that the self-consistent input and output correlations are suppressed. Perturbations of this structure in the local input typically lead to an increase in correlations [14]. In this study, we demonstrate this by replaying spiking activity after randomization of spike times, i.e., by replacing the time of each input spike by a random number uniformly drawn from the full emulation time interval $[0, T)$ (RAND case). However, even subtle modifications of input spike trains, such as random jitter of spike times by few milliseconds, lead to an increase of correlations. On the neuromorphic hardware, replay of spike trains is not entirely reproducible (see Section 2.4). Hence, spike-train correlations measured in the $\text{FB}_{\text{replay}}$ mode are slightly larger than in the FB case. We would expect the same effect on the input side (free membrane potentials). Due to hardware limitations, however, we can measure input correlations only in replay mode ($\text{FB}_{\text{replay}}$ or RAND), but not in the fully connected network (FB). Therefore, all reported input correlations are likely to be slightly overestimated. In conventional network simulations, we mimicked the effect of unreliable replay by input-spike jittering and, indeed, find a gradual increase in input and output correlations (data not shown). Despite the imperfect replay of input spikes, the decorrelation effect is clearly visible in hardware emulations, both on the input and on the output side. The reproducibility of emulations on neuromorphic hardware could be improved by stabilizing the environment of the system, e.g., the chip temperature or the support electronics (under development). Analog hardware, however, will never reach the level of reproducibility of digital computers. But note that, similar to analog hardware, biological neurons exhibit a considerable amount of trial-to-trial variability, even under controlled in-vitro conditions [66]. So far, it is unclear how neuronal noise such as, for example, synaptic stochasticity (spontaneous postsynaptic events, stochastic spike transmission, synaptic failure [76]), affects correlations in recurrent neural circuits.

Although different *Spikey* chips exhibit different realizations of fixed-pattern noise, they show a comparable extent of heterogeneity and yield results which are qualitatively similar to those presented in this article (Supplements 7). In the uncalibrated state, correlations

are more sensitive to the specific realization of fixed-pattern noise and therefore vary more strongly across different chips (see (B) in Supplements Figure 6 and 7). For the same reason, the variance of correlations across network realizations is largest in the uncalibrated state. Note that the variance across different network realizations is larger than the variance across different trials, i.e., consecutive emulations of identical networks (compare Figure 8 to Supplements Figure 8).

We have shown that negative feedback in recurrent circuits can efficiently suppress correlations, even in highly heterogeneous systems such as the analog neuromorphic architecture *Spikey*. Correlations can be further reduced by minimizing the level of network heterogeneity. In this study, we reduced the level of heterogeneity through calibration of neuron parameters in the unconnected case (see Section 2.5). The calibration could, in principle, be improved by calibrating neuron (and possibly synapse) parameters in the full recurrent network. Such calibration procedures are however time consuming and cumbersome. In biological substrates, homeostasis mechanisms [35, 77] keep neurons in a responsive regime and reduce the level of firing-rate heterogeneity in a self-regulating manner. Future neuromorphic devices could mimic this behavior, thereby reducing the necessity of time consuming calibration procedures.

For simplicity, this work focuses on purely inhibitory networks. On the one hand, this demonstrates that decorrelation by inhibitory feedback does not rely on a dynamical balance between excitation and inhibition (note that the external "excitatory" drive is constant in our model) [14, 50]. On the other hand, it remains unclear whether our results on the effect of heterogeneity generalize to systems with mixed excitatory-inhibitory coupling. Previous studies have shown that, for the homogeneous case, decorrelation by inhibitory feedback is a general phenomenon, which also occurs in excitatory-inhibitory networks, provided the overall inhibition is sufficiently strong (which is typically the case to ensure stability) [13, 14, 47, 50]. It is therefore likely that heterogeneity in excitatory-inhibitory networks plays a similar role as in purely inhibitory networks.

This study demonstrates that the *Spikey* system has matured to a level that permits its use as a tool for neuroscientific research. For the results presented in this study, we recorded in total 10^{11} membrane-potential and spike-train samples, representing more than 100 days of biological time. Due to the 10^4-fold acceleration of the *Spikey* chip, this corresponds to less than 15 minutes in the hardware-time domain. Interfacing the hardware system, however, reduces the acceleration to an approximately 50-fold speed-up (Figure 9). The translation between the network description and its hardware representation claims the majority of execution time, more than the network emulation and the transfer of data to and from the hardware system together. Especially encoding and decoding spike times on the host computer is computationally expensive. Obviously, the system could be optimized by processing data directly on the hardware or by choosing a data representation which is closer to the format used on the *Spikey* chip, but this would impair user-friendliness, and hence, the effectiveness of prototyping. While the *Spikey* system allows monitoring of the spiking activity of all neurons simultaneously, access to the membrane potentials is limited to a single (albeit arbitrary) neuron in each emulation run. Monitoring of membrane potentials of a population of n neurons therefore requires n repetitions of the same emulation. Extending the hardware system to enable access to the membrane potentials of at least two neurons simultaneously would allow a direct observation of input correlations in the intact network (and thereby avoid problems with replay reproducibility; see above) and reduce execution time (the *Spikey* chip itself permits recording of up to eight neurons in parallel, the support electronics, however, does not). While the *Spikey* system does not significantly outperform conventional computers in terms of computational power, emulations on this

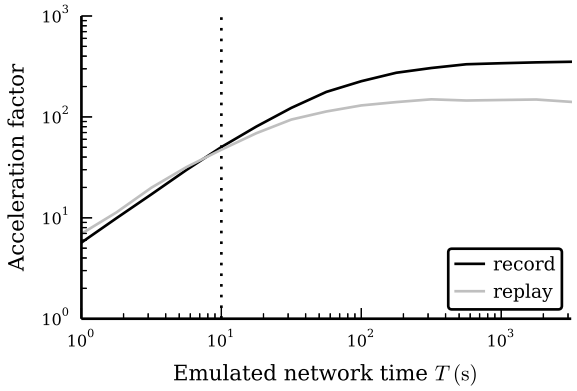


Figure 9: Acceleration factor as a function of emulated network time T for the record (black) and the replay case (gray). The acceleration factor is defined as the ratio between the emulated network time T (in biological time) and the execution time (wallclock time). In the record case, a network realization is generated on the host computer and uploaded to the chip. During the subsequent emulation, spike trains are recorded. In the replay case, spikes are replayed and the membrane potential of one neuron is recorded with full sampling frequency (9.6 kHz). The execution time covers the full data flow from a network description in `PyNN` to the emulation on the *Spikey* system and back to the network representation in `PyNN`. The time-averaged population firing rate is $\bar{r} = (25.0 \pm 0.4)\,\mathrm{s}^{-1}$. The vertical dashed line depicts the runtime used in this study. The hardware system has to be initialized once before usage (< 1 s), which is not considered here.

system are more energy efficient than simulations on conventional computers (Supplements 1). A substantial increase of computational power is expected for large systems exploiting the scalability of this technology without slow-down [78].

Functional neural architectures often rely on a stochastic dynamics of its constituents or on some form of background noise (see, e.g., [79, 49, 4]). Deterministic recurrent neural networks with inhibitory feedback could provide decorrelated noise to such functional networks, both in artificial as well as in biological substrates. In neuromorphic hardware applications, these "noise networks" could thereby replace conventional random-number generators and avoid a costly transmission of background noise from a host computer to the hardware substrate (which may be particularly relevant for mobile applications with low power consumption; see Supplements 1). It needs to be investigated, however, how well functional stochastic circuits perform in the presence of such network-generated noise.

Acknowledgments

We would like to thank Matthias Hock, Andreas Hartel, Eric Müller and Christoph Koke for their support in developing and building the *Spikey* system, as well as Mihai Petrovici and Sebastian Schmitt for fruitful discussions. This research is partially supported by the Helmholtz Association portfolio theme SMHB, the Jülich Aachen Research Alliance (JARA), EU Grant 269921 (BrainScaleS), and EU Grant 604102 (Human Brain Project, HBP).

References

[1] Shadlen, M. N., & Newsome, W. T. (1998). The variable discharge of cortical neurons: Implications for connectivity, computation, and information coding. *J. Neurosci. 18*(10), 3870–3896.

[2] Abbott, L. F., & Dayan, P. (1999). The effect of correlated variability on the accuracy of a population code. *Neural Comput. 11*, 91–101.

[3] Tripp, B., & Eliasmith, C. (2007). Neural populations can induce reliable postsynaptic currents without observable spike rate changes or precise spike timing. *Cereb. Cortex 17*(8), 1830–1840.

[4] Schmuker, M., Pfeil, T., & Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proc. Natl. Acad. Sci. USA 111*(6), 2081–2086.

[5] Cohen, M. R., & Maunsell, J. H. R. (2009). Attention improves performance primarily by reducing interneuronal correlations. *Nat. Neurosci. 12*, 1594–1600.

[6] Softky, W. R., & Koch, C. (1993). The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs. *J. Neurosci. 13*(1), 334–350.

[7] van Vreeswijk, C., & Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science 274*, 1724–1726.

[8] Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci. 8*(3), 183–208.

[9] Tetzlaff, T., Morrison, A., Geisel, T., & Diesmann, M. (2004). Consequences of realistic network size on the stability of embedded synfire chains. *Neurocomputing 58–60*, 117–121.

[10] Kriener, B., Tetzlaff, T., Aertsen, A., Diesmann, M., & Rotter, S. (2008). Correlations and population dynamics in cortical networks. *Neural Comput. 20*, 2185–2226.

[11] Tetzlaff, T., Rotter, S., Stark, E., Abeles, M., Aertsen, A., & Diesmann, M. (2008). Dependence of neuronal correlations on filter characteristics and marginal spike-train statistics. *Neural Comput. 20*(9), 2133–2184.

[12] Ecker, A. S., Berens, P., Keliris, G. A., Bethge, M., & Logothetis, N. K. (2010). Decorrelated neuronal firing in cortical microcircuits. *Science 327*(5965), 584–587.

[13] Renart, A., De La Rocha, J., Bartho, P., Hollender, L., Parga, N., Reyes, A., & Harris, K. D. (2010). The asynchronous state in cortical circuits. *Science 327*, 587–590.

[14] Tetzlaff, T., Helias, M., Einevoll, G., & Diesmann, M. (2012). Decorrelation of neural-network activity by inhibitory feedback. *PLoS Comput. Biol. 8*(8), e1002596.

[15] Griffith, J. S., & Horn, G. (1966). An analysis of spontaneous impulse activity of units in the striate cortex of unrestrained cats. *J. Physiol. (Lond.) 186*(3), 516–534.

[16] Koch, K. W., & Fuster, J. M. (1989). Unit activity in monkey parietal cortex related to haptic perception and temporary memory. *Exp. Brain Res. 76*(2), 292–306.

[17] Shafi, M., Zhou, Y., Quintana, J., Chow, C., Fuster, J., & Bodner, M. (2007). Variability in neuronal activity in primate cortex during working memory tasks. *Neuroscience 146*(3), 1082–1108.

[18] Hromadka, T., DeWeese, M. R., & Zador, A. M. (2008). Sparse representation of sounds in the unanesthetized auditory cortex. *PLoS Biol. 6*(1).

[19] OĆonnor, D. H., Peron, S. P., Huber, D., & Svoboda, K. (2010). Neural activity in barrel cortex underlying vibrissa-based object localization in mice. *Neuron 67*(6), 1048–1061.

[20] Roxin, A., Brunel, N., Hansel, D., Mongillo, G., & van Vreeswijk, C. (2011). On the distribution of firing rates in networks of cortical neurons. *J. Neurosci. 31*(45), 16217–16226.

[21] Buzsaki, G., & Mizuseki, K. (2014). The log-dynamic brain: how skewed distributions affect network operations. *Nat. Rev. Neurosci. 15*, 264–278.

[22] Song, S., Sjöström, P., Reigl, M., Nelson, S., & Chklovskii, D. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biol. 3*(3), e68.

[23] Lefort, S., Tomm, C., Sarria, J.-C. F., & Petersen, C. C. H. (2009). The excitatory neuronal network of the C2 barrel column in mouse primary somatosensory cortex. *Neuron 61*, 301–316.

[24] Koulakov, A. A., Hromadka, T., & Zador, A. M. (2009). Correlated connectivity and the distribution of firing rates in the neocortex. *J. Neurosci. 29*(12), 3685–3694.

[25] Avermann, M., Tomm, C., Mateo, C., Gerstner, W., & Petersen, C. (2012). Microcircuits of excitatory and inhibitory neurons in layer 2/3 of mouse barrel cortex. *J. Neurophysiol. 107*(11), 3116–3134.

[26] Ikegaya, Y., Sasaki, T., Ishikawa, D., Honma, N., Tao, K., Takahashi, N., Minamisawa, G., Ujita, S., & Matsuki, N. (2013). Interpyramid spike transmission stabilizes the sparseness of recurrent network activity. *Cereb. Cortex 23*(2), 293–304.

[27] Roxin, A. (2011). The role of degree distribution in shaping the dynamics in networks of sparsely connected spiking neurons. *Front. Comput. Neurosci. 5*(8).

[28] Kuhn, A., Aertsen, A., & Rotter, S. (2004). Neuronal integration of synaptic input in the fluctuation-driven regime. *J. Neurosci. 24*(10), 2345–2356.

[29] Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., & Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci. 5*(73).

[30] Mead, C. (1990). Neuromorphic electronic systems. *Proc. IEEE 78*(10), 1629–1636.

[31] FACETS (2010). Fast Analog Computing with Emergent Transient States, project website. Available at: `http://www.facets-project.org`.

[32] BrainScaleS (2014). Project website. Available at: `http://www.brainscales.eu`.

[33] Human Brain Project (2014). Project website. Available at: `http://www.humanbrainproject.eu`.

[34] Stein, R. B., Gossen, E. R., & Jones, K. E. (2005). Neuronal variability: Noise or part of the signal? *Nat. Rev. Neurosci. 6*, 389–397.

[35] Marder, E., & Goaillard, J.-M. (2006). Variability, compensation and homeostasis in neuron and network function. *Nat. Rev. Neurosci. 7*(7), 563–574.

[36] Van Vreeswijk, C., & Sompolinsky, H. (1998). Chaotic balanced state in a model of cortical circuits. *Neural Comput. 10*(6), 1321–1371.

[37] Tsodyks, M., Mitkov, I., & Sompolinsky, H. (1993). Pattern of synchrony in inhomogeneous networks of oscillators with pulse interactions. *Phys. Rev. Lett. 71*(8).

[38] Golomb, D., & Rinzel, J. (1993). Dynamics of globally coupled inhibitory neurons with heterogeneity. *Phys. Rev. E 48*(6), 4810–4814.

[39] Neltner, L., Hansel, D., Mato, G., & Meunier, C. (2000). Synchrony in heterogeneous networks of spiking neurons. *Neural Comput. 12*(7), 1607–1641.

[40] Denker, M., Timme, M., Diesmann, M., Wolf, F., & Geisel, T. (2004). Breaking synchrony by heterogeneity in complex networks. *Phys. Rev. Lett. 92*(7), 074103–1–074103–4.

[41] Mejias, J. F., & Longtin, A. (2012). Optimal heterogeneity for coding in spiking neural networks. *Phys. Rev. Lett. 108*.

[42] Chelaru, M. I., & Dragoi, V. (2008). Efficient coding in heterogeneous neuronal populations. *Proc. Natl. Acad. Sci. USA 105*(42).

[43] Padmanabhan, K., & Urban, N. N. (2010). Intrinsic biophysical diversity decorrelates neuronal firing while increasing information content. *Nat. Neurosci. 13*(10), 1276–1282.

[44] Yim, M. Y., Aertsen, A., & Rotter, S. (2013). Impact of intrinsic biophysical diversity on the activity of spiking neurons. *Phys. Rev. E 87*.

[45] Lengler, J., Jug, F., & Steger, A. (2013). Reliable neuronal systems: The importance of heterogeneity. *PLoS One*.

[46] Tripathy, S. J., Padmanabhan, K., Gerkin, R. C., & Urban, N. N. (2013). Intermediate intrinsic diversity enhances neural population coding. *Proc. Natl. Acad. Sci. USA 110*(20), 8248–8253.

[47] Bernacchia, A., & Wang, X.-J. (2013). Decorrelation by recurrent inhibition in heterogeneous neural circuits. *Neural Comput. 25*(7), 1732–1767.

[48] Schemmel, J., Grübl, A., Meier, K., & Müller, E. (2006). Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, pp. 1–6. IEEE Press.

[49] Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., Schmuker, M., Brüderle, D., Schemmel, J., & Meier, K. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci. 7*(11).

[50] Helias, M., Tetzlaff, T., & Diesmann, M. (2014). The correlation structure of local cortical networks intrinsically results from recurrent dynamics. *PLoS Comput. Biol. 10*(1), e1003428.

[51] Badoni, D., Giulioni, M., Dante, V., & Del Giudice, P. (2006). An aVLSI recurrent network of spiking neurons with reconfigurable and plastic synapses. In *Proceedings of the 2006 International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, pp. 4. IEEE Press.

[52] Hafliger, P. (2007). Adaptive WTA with an analog VLSI neuromorphic learning chip. *IEEE Trans. Neural Netw. 18*(2), 551–572.

[53] Vogelstein, R., Mallik, U., Vogelstein, J., & Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw. 18*(1), 253–265.

[54] Indiveri, G., Chicca, E., & Douglas, R. (2009). Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition. *Cognitive Computation 1*(2), 119–127.

[55] Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gomez-Rodriguez, F., Camunas-Mesa, L., Berner, R., Rivas-Perez, M., Delbruck, T., Liu, S.-C., Douglas, R., Hafliger, P., Jimenez-Moreno, G., Ballcels, A., Serrano-Gotarredona, T., Acosta-Jimenez, A., & Linares-Barranco, B. (2009). CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory - processing - learning - actuating system for high-speed visual object recognition and tracking. *IEEE Trans. Neural Netw. 20*(9), 1417–1438.

[56] Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., & Degnan, B. (2013). A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Trans. Biomed. Circuits Syst. 7*(1), 71–81.

[57] Benjamin, B., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A., Bussat, J., Alvarez-Icaza, R., Arthur, J., Merolla, P., & Boahen, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE 102*(5), 699–716.

[58] Renaud, S., Tomas, J., Lewis, N., Bornat, Y., Daouzli, A., Rudolph, M., Destexhe, A., & Saïghi, S. (2010). PAX: A mixed hardware/software simulation platform for spiking neural networks. *Neural Networks 23*(7), 905–916.

[59] Yu, T., & Cauwenberghs, G. (2010). Analog VLSI biophysical neurons and synapses with programmable membrane channel kinetics. *IEEE Trans. Biomed. Circuits Syst. 4*(3), 139–148.

[60] Davison, A., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., & Yger, P. (2009). PyNN: a common interface for neuronal network simulators. *Front. Neuroinformatics 2*(11).

[61] Brüderle, D., Müller, E., Davison, A., Muller, E., Schemmel, J., & Meier, K. (2009). Establishing a novel modeling tool: A Python-based interface for a neuromorphic hardware system. *Front. Neuroinformatics 3*(17).

[62] Kaplan, B., Brüderle, D., Schemmel, J., & Meier, K. (2009). High-conductance states on a neuromorphic hardware system. In *Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN)*, Atlanta, pp. 1524–1530. IEEE Press.

[63] Bill, J., Schuch, K., Brüderle, D., Schemmel, J., Maass, W., & Meier, K. (2010). Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity. *Front. Comput. Neurosci. 4*(129).

[64] Brüderle, D., Bill, J., Kaplan, B., Kremkow, J., Meier, K., Müller, E., & Schemmel, J. (2010). Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, France. IEEE Press.

[65] Jahnke, S., Memmesheimer, R.-M., & Timme, M. (2009). How chaotic is the balanced state? *Front. Comput. Neurosci. 3*(13), 1–15.

[66] Mainen, Z. F., & Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science 268*, 1503–1506.

[67] De la Rocha, J., Doiron, B., Shea-Brown, E., Kresimir, J., & Reyes, A. (2007). Correlation between neural spike trains increases with firing rate. *Nature 448*(16), 802–807.

[68] Wiechert, M. T., Judkewitz, B., Riecke, H., & Friedrich, R. W. (2010). Mechanisms of pattern decorrelation by recurrent neuronal circuits. *Nat. Neurosci. 13*, 1003–1010.

[69] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). Cambridge University Press.

[70] Pfeil, T., Scherzer, A.-C., Schemmel, J., & Meier, K. (2013). Neuromorphic learning towards nano second precision. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–5. IEEE Press.

[71] Bair, W., Zohary, E., & Newsome, W. (2001). Correlated firing in Macaque visual area MT: time scales and relationship to behavior. *J. Neurosci. 21*(5), 1676–1697.

[72] Kohn, A., & Smith, M. A. (2005). Stimulus dependence of neuronal correlations in primary visual cortex of the Macaque. *J. Neurosci. 25*(14), 3661–3673.

[73] Moreno-Bote, R., & Parga, N. (2006). Auto- and crosscorrelograms for the spike response of leaky integrate-and-fire neurons with slow synapses. *Phys. Rev. Lett. 96*, 028101.

[74] Helias, M., Tetzlaff, T., & Diesmann, M. (2013). Echoes in correlated neural systems. *New J. Phys. 15*, 023002.

[75] Grytskyy, D., Tetzlaff, T., Diesmann, M., & Helias, M. (2013). A unified view on weakly correlated recurrent networks. *Front. Comput. Neurosci. 7*, 131.

[76] Ribrault, C., Sekimoto, K., & Triller, A. (2011). From the stochasticity of molecular processes to the variability of synaptic transmission. *Nat. Rev. Neurosci. 12*, 375–387.

[77] O'Leary, T., Williams, A. H., Franci, A., & Marder, E. (2014). Cell types, network homeostasis, and pathological compensation from a biologically plausible ion channel expression model. *Neuron 82*(4), 809–821.

[78] Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, pp. 1947–1950. IEEE Press.

[79] Maass, W. (2014). Noise as a resource for computation and learning in networks of spiking neurons. *Proc. IEEE 102*(5), 860–880.

[80] Nordlie, E., Gewaltig, M.-O., & Plesser, H. E. (2009). Towards reproducible descriptions of neuronal network models. *PLoS Comput. Biol. 5*(8), e1000456.

[81] Sharp, T., Galluppi, F., Rast, A., & Furber, S. (2012). Power-efficient simulation of detailed cortical microcircuits on SpiNNaker. *J. Neurosci. Methods 210*(1), 110–118.

[82] Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia 2*(4), 1430.

Appendix A: Network description

See Table 1 and 2.

Appendix B: Description of data analysis

See Table 3 and 4.

A	**Model summary**
Populations	One (inhibitory)
Topology	-
Connectivity	Random convergent connections (fixed in-degree)
Neuron model	Leaky integrate-and-fire (LIF), fixed firing threshold, fixed absolute refractory time
Channel models	-
Synapse model	Exponentially decaying conductances, fixed delays
Plasticity	-
External input	Resting potential higher than threshold (= constant current) ($E_\mathrm{l} > \Theta$)
Measurements	Spikes and membrane potentials
Other	No autapses, no multapses

B	**Populations**	
Name	**Elements**	**Size**
I	LIF neuron	N

C		**Connectivity**
Source	**Target**	**Pattern**
I	I	Random convergent connect, in-degree K

D	**Neuron and synapse model**
Type	Leaky integrate-and-fire, exponential conductances
Subthreshold dynamics	Subthreshold dynamics ($t \notin (t^*, t^* + \tau_\mathrm{ref})$): $C_\mathrm{m} \frac{\mathrm{d}}{\mathrm{d}t} v(t) = -g_\mathrm{l}(v(t) - E_\mathrm{l}) - g_\mathrm{syn}(t)(v(t) - E_\mathrm{inh})$ Reset and refractoriness ($t \in (t^*, t^* + \tau_\mathrm{ref})$): $v(t) = v_\mathrm{reset}$ This model is emulated by analog circuitry on the *Spikey* chip [29].
Conductance dynamics	For each presynaptic spike at time t^* ($t > t^* + d$): $g_\mathrm{syn}(t) \approx J \exp(-\frac{t - t^* - d}{\tau_\mathrm{syn}})$, where $J = w_\mathrm{hw} g_\mathrm{max}$ This model is emulated by analog circuitry on the *Spikey* chip [49].
Spiking	If $v(t^*-) < \Theta \wedge v(t^*+) \geq \Theta$: emit spike with time stamp t^*

Table 1: Decription of the network model (according to [80]).

B	**Populations**	
Name	**Values**	**Description**
N	$\{96, 112, 128, 144, 160, 176, \mathbf{192}\}$	network size
C	**Connectivity**	
Name	**Values**	**Description**
K	15	number of presynaptic partners
D	**Neuron**	
Name	**Values**	**Description**
C_m	0.2 nF	membrane capacitance
τ_ref	1 ms	refractory period
v_reset	−80 mV	reset potential
E_l	−52 mV	resting potential
Θ	−62 mV	firing threshold
g_l	calibrated	leak conductance
E_inh	−80 mV	inhibitory reversal potential
D	**Synapse**	
Name	**Values**	**Description**
g_max	in the order of 1 nS	conductance amplitude
τ_syn	$\ll \frac{C_\mathrm{m}}{g_\mathrm{l}}$	conductance time constant
w_hw	3	synaptic weight (in hardware values $\in [0, 15]$)
d	≈ 1.3 ms	synaptic delay
Other	**Software**	
Name	**Values**	**Description**
	67bc2eec	git revision SpikeyHAL
	e793bb97	git revision pyNN
	d948716a	git revision vmodule

Table 2: Parameter values for the network model described in Table 1. Bold numbers indicate default values. Leak conductances g_l are adjusted in the calibration process (see Section 2.5). Gray numbers indicate target values not considering fixed-pattern noise.

A	**Analysis measures**
Measure	**Details**
spike density	$\xi_i(t) = \sum_k \delta(t - t_i^k)$
spike train	$s_{\mathrm{i}}(t_k)$ = number of spikes of neuron i per bin $[k\Delta t, (k+1)\Delta t)$
population activity	$\bar{s}(t) = \frac{1}{N}\sum_i s_{\mathrm{i}}(t)$
time-averaged population activity	$\bar{r} = \langle \bar{s}(t) \rangle_t$
membrane potential	$v_{\mathrm{i}}(t_k)$ = membrane potential of neuron i at time step k
(finite time) Fourier transform	$X_i(f) = \mathfrak{F}[x_i(t)](f) = \int_0^T dt\, x_i(t) e^{-2\pi i f t}$ (with inverse $\mathfrak{F}^{-1}$)
(single unit) power spectrum	$A_i(f) = \frac{1}{T} X_i^*(f) X_i(f)$
population-averaged power spectrum	$A(f) = \frac{1}{N}\sum_i A_i(f)$
population power spectrum	$\bar{A}(f) = (\sum_i S_i^*(f))(\sum_j S_j(f))$
pairwise cross spectrum	$C_{ij} = \frac{1}{T} X_i^*(f) X_j(f), i \neq j$
population-averaged cross spectrum	$C(f) = \frac{1}{N(N-1)} \sum_{i \neq j} C_{ij}(f) \equiv \frac{1}{N(N-1)} (\bar{A}(f) - NA(f))$ (note: $C(f) \in \mathbb{R}$)
sliding window filter	$X(f) \to X(f) * H(f)$ with $H(f) = \frac{1}{f_1 - f_0} \Theta(f - f_0)\Theta(f_1 - f)$
coherence	$\kappa(f) = \frac{C(f)}{A(f)}$
low-frequency coherence	$\kappa_X = \frac{1}{f_{\max} - f_{\min}} \int_{f_{\min}}^{f_{\max}} df \kappa(f)$
pop.-averaged cross-correlation function	$c(\tau) = \frac{1}{N(N-1)} \sum_{i \neq j} \langle s_{\mathrm{i}}(t) s_{\mathrm{j}}(t+\tau) \rangle_t \equiv \mathfrak{F}^{-1}[C(f)](\tau)$
trial average	$\langle \dots \rangle_k$
time average	$\langle \dots \rangle_t$

Table 3: Summary of the data analysis. Here $i \in [1, N]$, $X \in S, V$.

A	**Analysis parameters**	
Parameter	**Description**	**Values**
Δt	bin size for spike trains	1 ms
Δt_{m}	bin size for membrane potential traces	0.52 ms
T_{warmup}	initial warmup time (not considered in analysis)	1 s
T	emulated network time (biol. time domain)	10 s
M	number of trials (different network realizations)	$\{50, \mathbf{100}\}$
ΔF	width of sliding window	1 Hz
$f_{\mathrm{min}}, f_{\mathrm{max}}$	interval boundaries for low-frequency coherence	0.1 Hz, 20 Hz
a	calibration state	$\{0, \frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}, \mathbf{1}\}$

Table 4: Summary of analysis parameters (default values in bold).

Supplements

Supplements 1 Power consumption

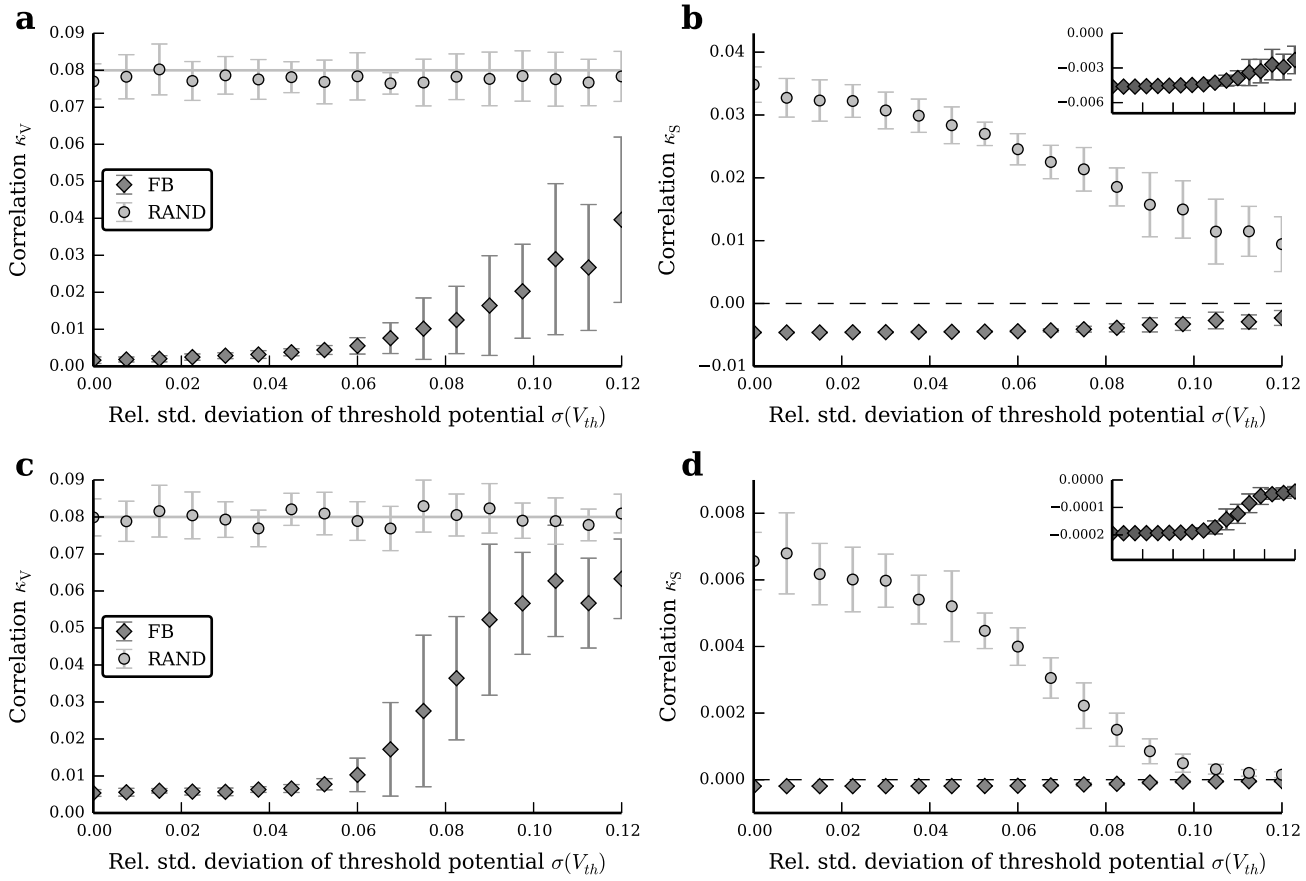
The *Spikey* system consumes approximately 6 W of power, and the chip itself less than 0.6 W. On the chip most power is consumed by digital communication infrastructure, which is not part of the neuromorphic network. In the following, we estimate the power consumption for a single synaptic event using the data set partly shown in Figure 5a. This emulation lasts $T = 10\,\mathrm{s}$ in biological time and generates approximately $48 \cdot 10^3$ spikes. Considering the acceleration of the hardware network (10^4) and the synapse count per neuron ($K = 15$), the system generates $720 \cdot 10^6$ synaptic events per second in hardware time. If we consider the total power consumption of the *Spikey* chip, the upper bound of energy consumed by each synaptic transmission will be approximately 1 nJ. Because these measurements include the communication infrastructure and other support electronics to observe spike times and membrane traces, the real energy consumption for synaptic transmissions is estimated to be approximately ten times smaller. Network simulations on conventional supercomputers a far less energy efficient and consume tens of µJ for each synaptic transmission [81].

Supplements 2 Modification of the bisection method

In each iteration of the bisection method that is used to calibrate the leak conductances of hardware neurons (Section 2.5), we evaluated the firing rate for each neuron by the median over $L = 25$ identical trials. However, if this measure is compared between consecutive identical iterations, temporal noise on time scales longer than the duration of one iteration may still lead to variability. In the original bisection method, the interval of possible solutions is halved after each iteration step [69]. To improve the convergence of this method in the context of our calibration we expanded the halved interval by 20% at both ends after each iteration. This prevents the algorithm to get stuck in an interval wrongly chosen by random fluctuations of the firing rates.

Supplements 3 Simulations with software

We validate our results by comparing them to simulations with software (NEST [82]). In these simulations we modulated the degree of heterogeneity by distributing the thresholds of all neurons according to a normal distribution with mean Θ and variance $\sigma(\Theta)$. Details about the network, neuron and synapse models and their parameters can be found in Supplements Table 1 and 2, respectively. The results are qualitatively the same compared to network emulations on the *Spikey* chip (compare Supplements Figure 1 to Figure 8). In the FB case, input correlations increase with network heterogeneity and spike-train correlations become less negative. In the RAND case, input correlations stay approximately constant, while output correlations decrease with the variance $\sigma(\Theta)$. This can be explained by the fact that, here, heterogeneity only affects the output spike times, and not the integrative properties of the neurons (see also Section 3.2 and [44]).



SUP. FIG. 1: Dependence of population-averaged input correlations **(a)** and **(c)**, and spike-train correlations **(b)** and **(d)** on the width of the threshold distribution for networks of two different sizes ($N = 192$ for (a) and (b), and $N = 4800$ for (c) and (d); both networks have a connectivity of approximately $\epsilon = 0.08$), for the FB (diamonds) and RAND (circles) case. Networks are simulated with NEST [82]. Symbols and error bars denote mean and standard deviation, respectively, across $M = 20$ (input correlations) and $M = 30$ (output correlations) network realizations (error bars are partly covered by markers). The gray curve in (a) and (c) depicts shared-input correlations in a homogeneous network (Equation 2). The black dashed line in (b) and (d) marks zero for orientation. Note that in simulations the $\text{FB}_{\text{replay}}$ is indentical to the FB case, and is hence not shown. Also note the different y-scales in panel (b) and (d).

Supplements 4 Linear model

We investigate the consistency of our results with a linear rate model that allows to numerically calculate the average correlations from a given connectivity matrix $\mathbf{W}$. The model is defined as (according to, e.g., [75])

$$\mathbf{r}(t) = (\mathbf{W}(\mathbf{r}+\mathbf{x}) * h)(t) \quad . \tag{S1}$$

Here, $\mathbf{r}(t)$ denotes the rate of the individual neurons and $\mathbf{x}(t)$ a Gaussian white noise input that is independent for each neuron. The linear filter kernel $h(t)$ depends on the details of the model, is not relevant in our calculation, and hence is not further specified, here. Equation S1 can be transformed to Fourier domain, where the input and output spectral matrices can be expressed by

$$\mathbf{C}_{\mathrm{RR}}(\omega) = T(\omega)T(\omega)^{\dagger} \quad , \tag{S2}$$

$$\mathbf{C}^{\mathrm{in}}_{\mathrm{RR}}(\omega) = \mathbf{W}\mathbf{C}_{\mathrm{RR}}(\omega)\mathbf{W}^{T} \quad , \tag{S3}$$

with $T(\omega) = (1 - H(\omega)\mathbf{W})^{-1}$ [75]. In the RAND case, the linear equation for the rate of the (unconnected) neurons reads

$$\mathbf{q}(t) = (\mathbf{W}(\tilde{\mathbf{r}}+\mathbf{x}) * h)(t) \quad , \tag{S4}$$

where $\tilde{\mathbf{r}}(t)$ has the same auto-correlations as $\mathbf{r}(t)$ but zero cross correlations, i.e., $\mathbf{C}_{\tilde{\mathrm{R}}\tilde{\mathrm{R}}} = \mathrm{diag}(\mathbf{C}_{\mathrm{RR}})$, since the randomization of spike times removes all spatio-temporal correlations. According to Tetzlaff et al. [14] spectral matrices in the RAND case are given by

$$\mathbf{C}^{\mathrm{in}}_{\mathrm{QQ}}(\omega) = \mathbf{W}\mathbf{C}_{\tilde{\mathrm{R}}\tilde{\mathrm{R}}}\mathbf{W}^{T} \quad , \tag{S5}$$

$$\mathbf{C}_{\mathrm{QQ}}(\omega) = |H(\omega)|^2(\mathbf{C}^{\mathrm{in}}_{\mathrm{QQ}} + \rho) \quad . \tag{S6}$$

We calculate the population-averaged power- and cross-spectra from the full matrices:

$$\bar{A}_{\mathrm{X}}(\omega) = \frac{1}{N}\sum_i C_{\mathrm{XX},ii} \quad , \tag{S7}$$

$$\bar{C}_{\mathrm{XX}}(\omega) = \frac{1}{N(N-1)}\sum_{i\neq j} C_{\mathrm{XX},ij} \quad . \tag{S8}$$

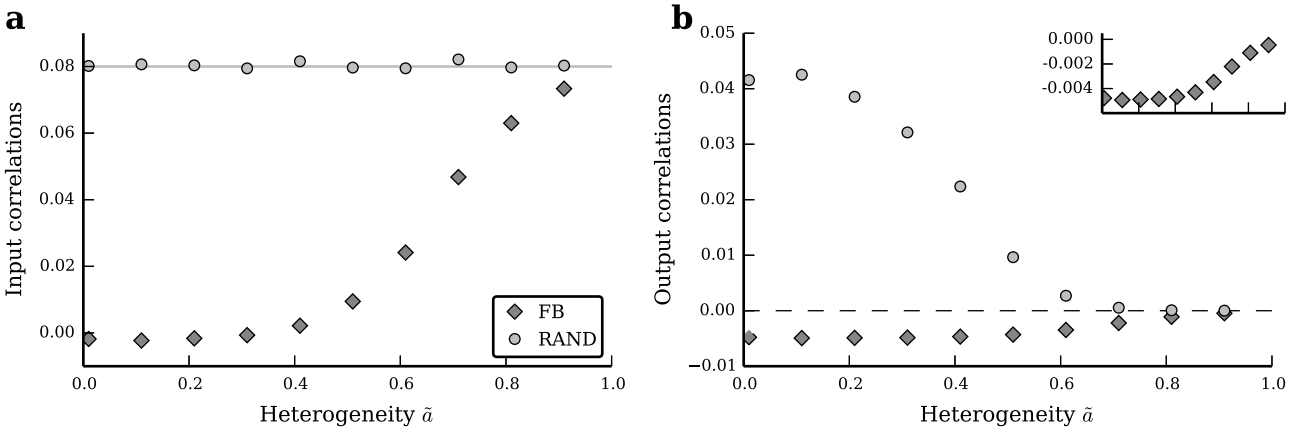
Here, $X \in \{R, Q\}$ denotes the FB and RAND case, respectively. The low frequency coherence is the cross-spectra normalized by the power spectra:

$$\kappa_X(0) = \frac{\bar{C}_{\mathrm{XX}}(0)}{\bar{A}_{\mathrm{X}}(0)} \quad . \tag{S9}$$

Note that in the linear model we are actually taking the zero frequency coherence.

As in the spiking model, we consider a sparse network, i.e., we randomly choose for each neuron $i \in [1, N]$ an identical number of presynaptic partners ($K = 15$). In the linear model we do not consider a distribution of non-zero effective weights. Instead, each realized connection is assigned the same weight value $-w$. To mimic the effect of calibration we vary the absolute value of the effective weight by scaling the weights of the non-zero connections with a sigmoidal function of $\tilde{a} \in [0, 1]$:

$$\tilde{w} = \frac{1}{1 + e^{10\times(\tilde{a}-0.5)}} w \quad . \tag{S10}$$



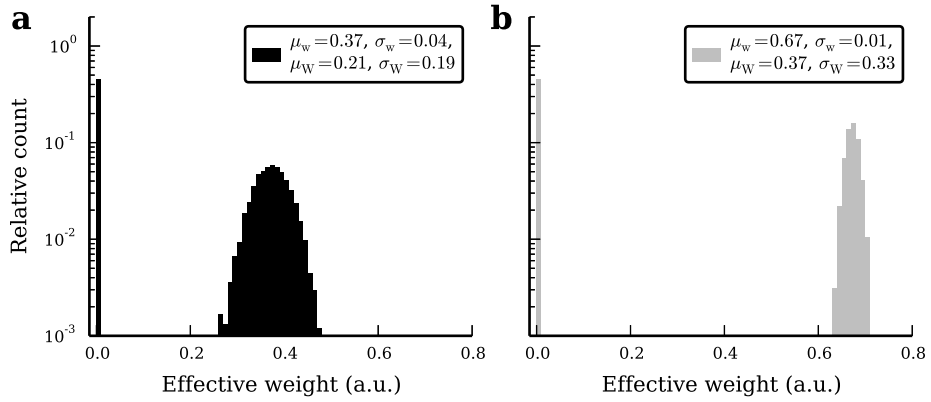
SUP. FIG. 2: Dependence of population-averaged input **(a)** and output correlations **(b)** on the heterogeneity of the effective weight matrix, for the FB (diamonds) and RAND (circles) case. The gray curve in (a) depicts shared-input correlations in a homogeneous network (Equation 2). The black dashed line in (b) marks zero correlations for orientation. Note that in the linear model the $\text{FB}_{\text{replay}}$ is identical to the FB case, and is hence not shown.

This procedure changes the variance of the weight matrix [47] and hence $\tilde{a}$ is denoted the *heterogeneity* of the network. More homogeneous (heterogeneous) networks have a larger (smaller) effective weights and hence a stronger (weaker) feedback loop. We obtain qualitatively similar results as we observe on the *Spikey* chip (compare Supplements Figure 2 to Figure 8). Correlations in the RAND case decrease, while correlations in the FB case increase with network heterogeneity, i.e., with the variance of the effective weight matrix.

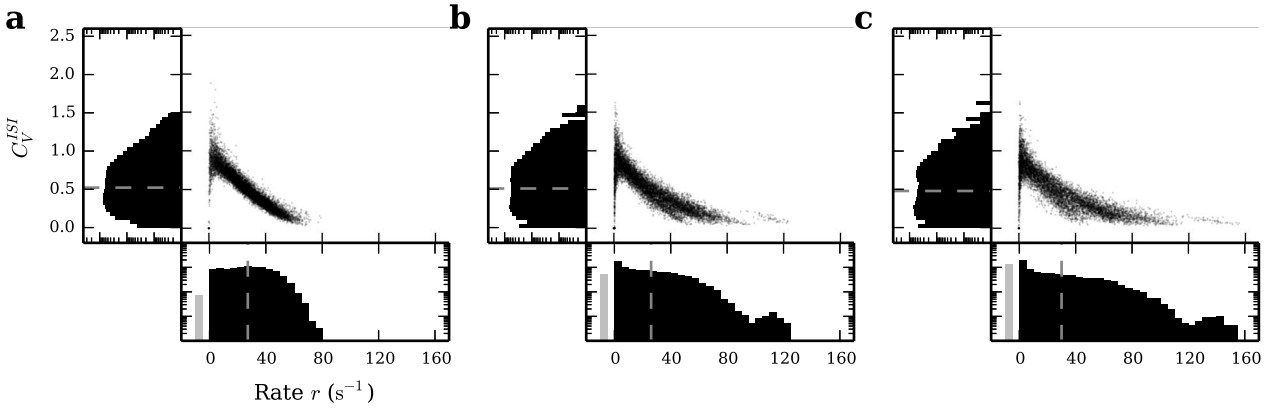
In Supplements Figure 3 we illustrate, how in a sparse network the variance of the weight matrix can increase, although the distribution of non-zero weights narrows. The standard deviation σ_{w} of a distribution of non-zero weights with mean μ_{w} is (in this example) smaller than the standard deviation σ_{W} of the full effective weight matrix, due to the sparseness of the matrix (Supplements Figure 3a; here, we chose $\epsilon = 0.8$). If we, at the same time, increase the mean μ_{w} and decrease the standard deviation σ_{w} of non-zero weights, the standard deviation of the weight matrix σ_{W} can increase significantly (Supplements Figure 3). While the distribution of effective weights is broadened, the mean is decreased playing the more significant role for the size of correlations. This observation could explain the decrease of correlations with increasing calibration.

Supplements 5 Firing statistics in the RAND case

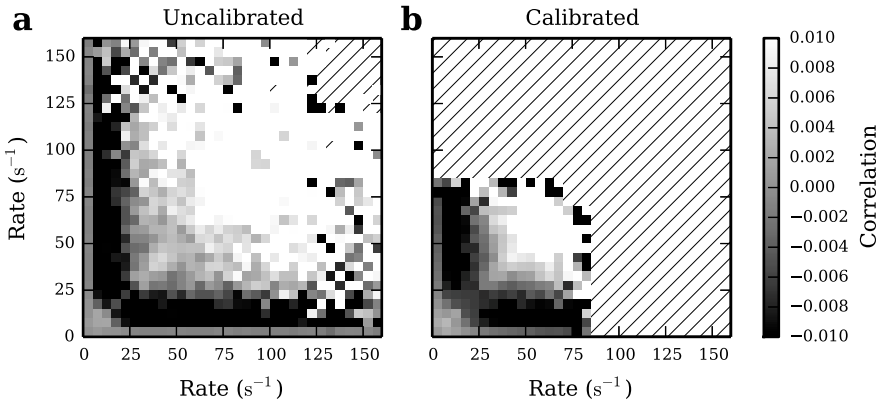
The firing rate distributions in the RAND case are similar to those in the FB scenario (compare Supplements Figure 4 to Figure 7d–f). They are narrower for more homogeneous networks. Nevertheless, former inactive neurons in the FB case have a higher probability to fire in the RAND case, because the temporal fluctuations of their membrane potentials increase due to higher correlations in their input (Figure 5e). Neurons with firing rates above average are barely affected by this effect, because they are strongly driven by the constant current influx, and hence show similar firing rates than in the FB scenario. The regularity of firing increases for the RAND compared to the FB case, also due to stronger fluctuations of the input.



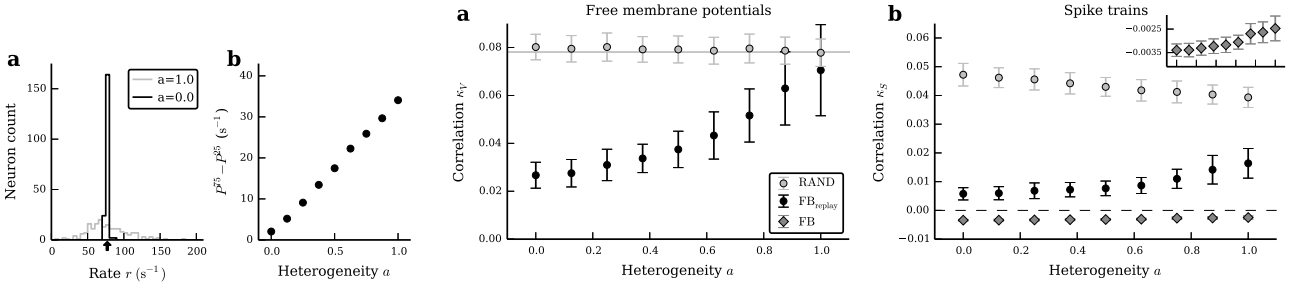
SUP. FIG. 3: Distributions of effective weights in a sparse network. Different values for the mean μ_w and standard deviation σ_w of the distribution of non-zero weights in **(a)** and **(b)**, respectively, result in different values for the mean μ_W and standard deviation σ_W of the effective weight matrix.



SUP. FIG. 4: **(a–c)** Like Figure 7d–f, but for the RAND case. The mean of firing rate ($27.2\,s^{-1}$, $26.0\,s^{-1}$, $30.0\,s^{-1}$) and C_V^{ISI} distributions (0.52, 0.51, 0.48) are marked with dashed lines.



SUP. FIG. 5: Pairwise spike-train correlations for all pairs of neurons for $M = 100$ trials, sorted by the rate of the respective neurons. Diagonal stripes indicate that no data was available. Here $C_{ij}(f) \in \mathbb{C}$, but we only consider the real part of the cross spectrum to calculate the low-frequency coherence: $\kappa_{ij} = \frac{1}{f_{\max} - f_{\min}} \int_{f_{\min}}^{f_{\max}} \frac{\mathfrak{Re}(C(f))}{A(f)}$.



(A) Calibration (B) Correlation

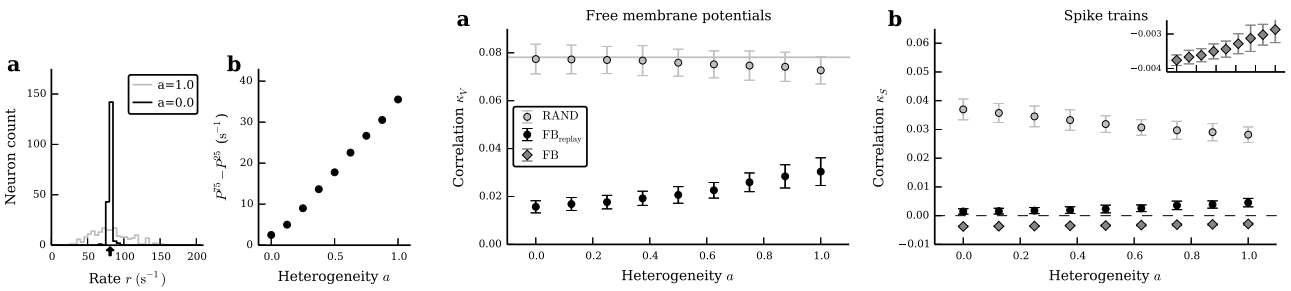
SUP. FIG. 6: Like chip 1 in Figure 4 and 8, but for chip 2 in (A) and (B), respectively.

Supplements 6 Correlation matrix

In addition to the population-averaged measures from the main manuscript, we also calculated the pairwise correlations for each pair i, j of neurons with $i \neq j \in [1, N]$, and ordered these by the time-averaged rate of the corresponding neurons (Supplements Figure 5). This reveals a dependence of the pairwise correlation on the rate of the respective neurons. If both neurons fire at low rate (here $< 5\,\mathrm{s}^{-1}$), correlations will be close to zero similar to the results in [67]. For high rates (here $> 25\,\mathrm{s}^{-1}$) we find mostly positive correlations. However, if both neurons fire at intermediate rates, the activity of neurons will be anti-correlated, and will suppress positive shared-input correlations. After calibration, the amount of neurons firing at intermediate rates increases, and hence shared-input correlations are suppressed by more neurons (Supplements Figure 5, Figure 7).

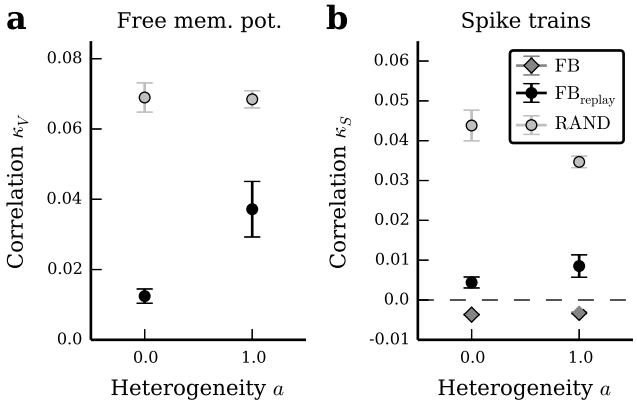
Supplements 7 Results for different *Spikey* chips

The experimental protocol presented in the main text was used for two additional *Spikey* chips. Different chips show different realizations of fixed-pattern noise, and hence calibration was repeated for each chip (Figure 4, Supplements Figure 6A and 7A). In the calibrated state free membrane potentials (and spike trains) are decorrelated by inhibitory



(A) Calibration (B) Correlation

SUP. FIG. 7: Like Supplements Figure 6, but for chip 3.



SUP. FIG. 8: Like Figure 8 and for the same chip, but for $L = 20$ trials of one network realization.

feedback for all chips (Figure 8, Supplements Figure 6B and 7B). However, the more uncalibrated the system is, the more the results differ between chips, which is likely to be caused by different extents of intrinsic fixed-pattern noise. This is most pronounced for chip 2 (Supplements Figure 6B), where the input correlations of the $\mathrm{FB}_{\mathrm{replay}}$ case reaches that of the RAND scenario for the uncalibrated state, which means that the input of neurons is not decorrelated by the inhibitory feedback anymore.

Supplements 8 Reproducibility of networks with intact feedback

We measured the variance of free membrane potential and spike train correlations over several trials for a single network realization (Supplements Figure 8). This variance is smaller than the variance we observe over different network realizations (compare to Figure 8 and Supplements Figure 6 and 7), which indicates that the latter is mostly caused by the different connectivity, not by trial-to-trial variability. Note that the variability between trials of networks with intact feedback is likely to be larger than between replays of network activity as shown in Figure 3, because network dynamics may be chaotic. The data shown in Supplements Figure 8 has to be interpreted with care, because the reproducibility of only a single network realization is considered. For different realizations, the variance may change.

A	**Model summary**
Populations	One (inhibitory)
Topology	-
Connectivity	Random convergent connections (fixed in-degree)
Neuron model	Leaky integrate-and-fire (LIF), fixed firing threshold, fixed absolute refractory time
Channel models	-
Synapse model	Exponentially decaying currents, fixed delays
Plasticity	-
External input	Resting potential higher than threshold (= constant current) ($E_\mathrm{l} > \Theta$)
Measurements	Spikes and membrane potentials
Other	No autapses, no multapses

B	**Populations**	
Name	**Elements**	**Size**
I	LIF neuron	N

C		**Connectivity**
Source	**Target**	**Pattern**
I	I	Random convergent connect, in-degree K

D	**Neuron and synapse model**
Type	Leaky integrate-and-fire, exponential currents
Subthreshold dynamics	Subthreshold dynamics ($t \notin (t^*, t^* + \tau_\mathrm{ref})$): $C_\mathrm{m} \frac{\mathrm{d}}{\mathrm{d}t} v(t) = -g_\mathrm{l}(v(t) - E_\mathrm{l}) + I_\mathrm{syn}(t)$ Reset and refractoriness ($t \in (t^*, t^* + \tau_\mathrm{ref})$): $v(t) = v_\mathrm{reset}$
Current dynamics	$\tau_\mathrm{syn} \frac{\mathrm{d}}{\mathrm{d}t} I_\mathrm{syn}(t) = -I_\mathrm{syn}(t) + \sum_{i,k} J\delta(t - t_i^k)$ Here the sum over i runs over all presynaptic neurons and the sum over k over all spiketimes of the respective neuron i
Spiking	If $v(t^*-) < \Theta \wedge v(t^*+) \geq \Theta$: emit spike with time stamp t^*

SUP. TABLE 1: Description of the network model (according to [80]).

B	**Populations**	
Name	**Values**	**Description**
N	$\{192, 4800\}$	network size
C	**Connectivity**	
Name	**Values**	**Description**
K	$\{15, 384\}$	number of presynaptic partners
D	**Neuron**	
Name	**Values**	**Description**
C_m	0.2 nF	membrane capacitance
τ_ref	0.1 ms	refractory period
v_reset	−80 mV	reset potential
E_l	−52 mV	resting potential
Θ	−62 mV	firing threshold
g_l	10 nS	leak conductance
D	**Synapse**	
Name	**Values**	**Description**
τ_syn	5 ms	conductance time constant
J	3 nA	synaptic weight
d	1.0 ms	synaptic delay

SUP. TABLE 2: Parameter values for the network model described in Table 1 (default values in bold).

Chapter 3

Discussion

In the following, we evaluate the potential of brain-inspired computing in general, and for the neuromorphic *Spikey* system in particular. Therefore, first, we briefly summarize the differences between neuromorphic and conventional computing (Section 3.1). Second, we discuss the key features of the used neuromorphic system in detail: The *Spikey* system is portable (Section 3.2), fast (Section 3.3), energy efficient (Section 3.4), and networks implemented on the chip are potentially plastic (Section 3.5) and robust (Section 3.6). Ideas for further possible studies and applications using the *Spikey* system, and future perspectives are given in Section 3.7.

3.1 Neuromorphic computing

Analog neuromorphic and digital conventional computing is different in certain aspects, as listed below. Note that in the following Sections 3.2 to 3.6 these aspects are discussed in detail on the example of the *Spikey* system.

For example, on conventional computers network models can be sized to more neurons than there are computing cores. Currently, networks are limited to 10^{13} synapses because of memory limitations (e.g., Kunkel et al., 2012, 2014), and the size of the human neocortex (10^{14} synapses have been reported by Pakkenberg et al., 2003) is not reachable, yet.

On hardware, in contrast, the network size is limited by the number of physical implementations of its constituents. The parallel arrangement of neurons and synapses, however, renders computation massively parallel.

In addition, on hardware neuron and synapse models are fixed, which allows to minimize both the complexity and size of their analog circuits, and hence enables for high integration densities. In contrast, for simulations with software, the flexibility of models is virtually unlimited. However, more complex models are usually computationally more expensive and hence elongate the runtime of simulations. By altering the source code, neuron and synapse models can be rapidly adapted to new hypothesis and experimental data. This is especially useful when it is still unclear how rich the dynamics of neurons and synapses have to be for the efficient implementation of brain-inspired algorithms. For example, how does computation benefit from features like ion channel dynamics (on molecular level, or on population level as used in the Hodgkin-Huxley model described in Section 1.1.1), multiple compartments, dendritic computation or plasticity mechanisms beyond STDP? The hardware design of neurons and synapses involves tradeoffs between model accuracy and costs in terms of chip area. In certain circumstances, a compact neuron or synapse circuit may deviate from the mathematical description, but may be closer to biological observations. For example, the step-wise rise of synaptic conductances in the LIF neuron

model (see g_{syn} in Section 1.1.2) is difficult to implement in hardware. A finite rise time, however, is easier to implement and is closer to biological measurements (Section 1.1.1) despite the more complex mathematical description and tractability. To verify the behavior of such implementations we propose the practice of preliminary studies to evaluate the performance of these implementations in the context of functional networks (see, e.g., Publication II).

On hardware, network models can not be paused and continued at any time, because the physical process underlying the network dynamics is continuous. In contrast, the intrinsic computation of physical models enables for a high acceleration at low chip area, and consequently high energy efficiency.

In software, practically each variable of the network can be accessed at any time, which simplifies the parallel acquisition of continuous variables, as, e.g., membrane potentials as used in Publication V, but potentially prolongs the simulation time. On hardware, the access to observables is typically limited. In case of the *Spikey* system the spike times of all neurons and membrane potential traces of few neurons can be accessed at the same time, while other variables, e.g., conductances are not directly observable. However, the bandwidth with which spike times can be recorded during accelerated network emulations is remarkable for the *Spikey* system (up to 300 million spikes per second in hardware time domain).

Prototype systems of analog neuromorphic hardware usually consist of highly custom and configurable network circuits. A significant step to an active community developing brain-inspired algorithms for such devices is to provide an abstract interface for the configuration and operation of these devices (e.g., Brüderle et al., 2009; Galluppi et al., 2010). Then, the standardized description language `PyNN` can be used to run networks on conventional computers (Davison et al., 2008) as well as on hardware systems, which simplifies the exploitation and comparison of different simulation and emulation platforms.

Both spatial and transient noise, i.e., variation between neuron and synapse parameters and between trials of network emulations, respectively, are abundant in biological systems (Mainen & Sejnowski, 1995; Stein et al., 2005; Marder & Goaillard, 2006; Shafi et al., 2007) and analog neuromorphic hardware (see, e.g., Brüderle, 2009; Neftci et al., 2011, and Section 3.6). The presence of noise facilitates the development of robust network models, but will also make the investigation of network dynamics more difficult, if repeatable experiment conditions are required. We propose the method of averaging out spatial and temporal variations by emulating different realizations of the same network model for all network implementations on hardware (as, e.g., applied in Publication III and V). Moreover, spatial variations can be reduced by calibration routines (for the *Spikey* system, see Publication I, V and Brüderle (2009)).

For the *Spikey* system, however, calibrations are not feasible for all parameters, because some parameters are shared between neurons and synapses, respectively. Nevertheless, this does not seem to limit the system's versatility, as demonstrated by the implementation of diverse network models throughout this work.

3.2 Portability and the roadmap towards practical applications

The physical dimensions of a computing substrate determines its suitability for portable applications outside the laboratory. In the following, the dimensions of the *Spikey* chip and system and possible reductions of these are discussed in the context of real world applications.

With a die size of 5 mm × 5 mm the *Spikey* chip has approximately the size of the cross-section of a pencil (Figure 1.5). Compared to a modern CPU[1] manufactured in a 22 nm process technology and consisting of 1.4 billion transistors, the *Spikey* chip is manufactured in a 180 nm process technology, and comprises complex circuits built from about 8 millions transistors. Migrating to a smaller process technology would reduce the area for digital circuits including synaptic weights, and would hence minimize the dimensions of the chip (see digital part and synapse arrays in Figure 1 of Publication I). The reduction of analog circuits, however, is likely to be less effective than for digital circuits, because their reduction affects the fixed-pattern noise and acceleration of the system (Hock, 2014a), and is hence limited by the effort for calibration and high-speed communication infrastructure.

Compared to the former backplane system that is mounted in a 19-inch rack, the USB version is substantially smaller, and consequently more attractive for mobile applications. For the latter, the support electronics is reduced in size and is optimized for single-chip operation. However, the application of this system in robotics is difficult because of the high acceleration of network emulations (for details, see Section 1.5.3). One approach to still exploit the acceleration in robotics is to modularize the task and serialize its emulation. This would bridge the gap between the different time scales and could improve the performance of emulations (for an example, see Section 3.7).

In order to run networks on the *Spikey* chip, is has to be powered and configured, which is provided by dedicated support electronics (Figure 1.5) and software (Brüderle et al., 2009). Additional features, like the recording of membrane potentials and debug functionality, are useful for the development of brain-inspired algorithms, but may be left out once the development of the algorithm is finished. The same customization process accounts for the network circuits on the chip. Once the connectivity and necessary configurability of a certain network model is found, the chip could be optimized for this scenario. For example, the removal of unused synapses (the connectivity of network models is usually sparse) or synaptic plasticity (for static networks, see Publication I, V, and also IV) could further reduce the chip dimensions.

Other hardware systems usually comprise chips of comparable size, but are larger on system level. For example, the wafer-scale system integrates many chips, has the size of a 19 inch rack and one wafer allows for the emulation of networks of up to 200000 neurons (Schemmel et al., 2010; Brüderle et al., 2011). Real time systems mostly have a volume of several liters, and a neuron count and configurability comparable to the *Spikey* system (Indiveri et al., 2009; Indiveri, 2012), or have a larger neuron count, but are limited to specific computational tasks (Serrano-Gotarredona et al., 2009).

Concluding, the *Spikey* chip and system has compact dimensions considering its remarkable configurability (Section 1.5.1), computational power (Section 3.3) and power-efficiency (Section 3.4). Consequently, the *Spikey* chip may be suitable for a coprocessor in portable systems like, e.g., in mobile phones or to a limited extend in robots (see above). To promote the development of algorithms for brain-inspired computing, the *Spikey* system could be distributed as a compact and powerful prototyping system for educational and/or scientific purposes.

3.3 Computational capability of the accelerated system

The advantage of an approximately 10^4-fold accelerated neuromorphic systems lies both in the short time-to-decision for single network emulations and the high frequency of emula-

[1] 3rd generation Intel® Core™ processor, also known as Ivy Bridge, with a die size of 160 mm^2

tions in the context of iterative operation (see, e.g., Publication IV). Although consecutive emulations that are independent of each other can be (virtually) parallelized on conventional computers, their execution on *Spikey* is more power efficient (Section 3.4). However, the massive parallelism and power efficiency is only guaranteed, if the network size does not exceed the number of neurons on a single chip (scaled by the maximum number of chips that can potentially be interconnected). Otherwise, the network has to be partitioned and emulated consecutively, if possible, losing the intrinsic acceleration of the chip due to the need for buffering and sorting the spiking activity as well as re-configuring the system (see also Müller, 2014).

Because the current hardware setup is of prototype nature, most of the pre- and post-processing of data takes place on the host computer, and hence the time to execute network emulations is much longer than the pure runtime of the hardware network. For future systems this overhead can be reduced by optimizing the software infrastructure and transferring computational expensive operations, as, e.g., the encoding and compression of spike data, to the FPGA or dedicated circuits. If only performance, but not user-friendliness and fast prototyping, is of concern, the `PyNN` layer could be omitted making the costly translation between voltage and time domains obsolete. Because the time required to configure the chip is constant, but the number of spikes fed into and received from the system usually grows linearly with runtime, the execution time increases with the runtime of the network (see Figure 9 in Publication V). Dependent on the task, currently, network emulations are effectively accelerated by a factor of 10 to 100 compared to biological real-time (for profilings, see Publication IV and V).

In order to come closer to the full acceleration of the analog hardware network, the amount of input and output data could be reduced. This can be achieved by networks that do not need external spike input (Publication V), or by networks for which only few neurons encode the final response (e.g., the readout of the classifier in Publication IV can be reduced to neurons of the association layer). Ideally, the *Spikey* chip directly interfaces spiking sensors and actuators, making the spike communication to the host computer obsolete.

At the moment, for small spike counts (emulations of few seconds) the configuration of analog parameters (for details, see Publication I) holds a substantial share of the execution time. These analog parameter cells need time to settle to their target value, which is the shorter the closer the last configuration was to this target value. By adapting the time reserved for parameter settling to the difference to its former value, the configuration of analog parameters could be substantially accelerated. In addition, so far, partial configuration of the chip is implemented only on a coarse level, which means, e.g., that blocks of digital synaptic weights can be written without the re-configuration of the analog parameters. An increase of the granularity of this process, e.g., by writing a subset of weights (only these that actually changed since the last emulation), would further improve the performance of emulations. Moreover, for emulations using the identical or a manageable set of stimulation patterns, all of these patterns could be buffered on the on-board memory in order to be recalled quickly without the necessity to transfer the data again. Similarly, configuration data can be buffered for later retrieval. Of course, the size of the on-board memory limits the size and number of stimulation patterns and configuration data that can be buffered. Such buffering is especially of advantage, if the sets of stimulation or configuration data is known in advance, as, e.g., for pattern recognition (Publication IV and liquid state machine in Publication I) or parameter sweeps (Publication V). Currently, this mechanism is not implemented, but instead identical stimulation patterns and configuration data have to be transferred again and again.

Most other hardware systems are not accelerated and operate in real time (for listing, see Section 1.2). This usually results in less computational power, but simplifies the interaction with the environment, e.g., for robotics.

3.4 Power efficiency of neuromorphic computation

For portable applications energy is usually a limiting factor, and hence data processing units should be power efficient, and ideally coolable by ambient temperature. The *Spikey* chip consumes approximately 0.6 W of power, and the system in total $P = 6\,\mathrm{W}$ (Publication V). Compared to other neuromorphic systems this system seems to be rather power consuming (e.g., Sharp et al., 2012). On the other hand, if one takes the system's acceleration into account and provided that static currents dominate the power consumption, as it is usually the case, the energy needed for a single synaptic transmission is on par with other low power systems (see Supplements in Publication V). This makes the system attractive to high-throughput applications like classification (see, e.g., Publication IV and liquid state machine in Publication I) and statistical analysis (see, e.g., Publication V) that can be efficiently implemented on the chip. Consequently, if intense computation is required and the chip is continuously used over time, this neuromorphic system is indeed suitable for mobile applications.

In general, neuromorphic systems outperform conventional computers or clusters of these in power efficiency (e.g., Arthur et al., 2012; Sharp et al., 2012; Benjamin et al., 2014). To compare systems operating at different speed, the power efficiency is compared by considering the energy for single synaptic transmissions. Note that the following numbers have to be treated with care, because the neuron, synapse and network model are different for each example. Additionally, the power efficiency of network simulations is dependent on the mean activity of the network model, which should be stable during these comparisons. In the following, we consider two scenarios:

First, we benchmarked the classifier network on the *Spikey* system (see also Publication IV) and on a modern desktop machine. Synaptic transmissions on the *Spikey* chip cost approximately six orders of magnitude less than on a conventional computer in terms of energy (compare *Spikey* chip to multi-threaded simulation with conductance-based synapses in Table 3.1). If we consider the power consumption of the whole *Spikey* system (excluding the host computer, for discussion see below), and the time to execute a network emulation including the full software stack, the gain in energy efficiency shrinks to less than two orders of magnitude. However, by optimizing the performance of the software interface and for long-lasting emulations (see Figure 9 in Publication V) the power efficiency of the *Spikey* system can be drastically improved. Additionally, the FPGA consumes the major share of power in the *Spikey* system, and could be replaced by dedicated circuits consuming less power. Software simulations can be accelerated by replacing the conductance-based synapse model by a current-based one, which allows for semi-analytical instead of purely numerical simulation techniques (Rotter & Diesmann, 1999). Note that the performance of simulations with software scales less than linear with the number of used threads, hence this network may not be executed efficiently in parallel on the desktop machine, like it is approximately the case for supercomputers (see e.g. Ananthanarayanan et al., 2009; Kunkel et al., 2014).

Second, supercomputers are investigated for different network models. Ananthanarayanan et al. (2009) showed a 1 s simulation of a network consisting of 1.617 billion neurons with $S = 8.87$ trillion synapses on a Blue Gene/P supercomputer with 147456 CPUs and 144 TB main memory. The execution of this network took $T = 3.41\,\mathrm{h}$ and resulted in a mean ac-

Simulation platform	Execution time (T)	Energy for syn. transm. (E)
Spikey chip (network time)	0.1 ms	230 pJ
Spikey system	(73 ± 1) ms	1.7 µJ
NEST with single thread (cond.-based syn.)	(595 ± 2) ms	191 µJ
NEST with 4 threads (cond.-based syn.)	(250 ± 2) ms	80 µJ
NEST with single thread (curr.-based syn.)	(142 ± 2) ms	46 µJ
NEST with 4 threads (curr.-based syn.)	(134 ± 2) ms	43 µJ

Table 3.1: Benchmark for classifier network using the simulator `NEST` (Gewaltig & Diesmann, 2007) and the *Spikey* (for details see Section 4.1). A simplified version of the classifier network consists of the identical number of inputs ($M = 60$), neurons ($N = 178$) and synapses ($S = 5520$), and is tuned towards the same average rate of inputs $R = (47.3 \pm 0.8)\,\mathrm{s}^{-1}$ and neurons $(11.5 \pm 0.9)\,\mathrm{s}^{-1}$ than in a typical use case of the original network (Schmuker, 2013): Note that the mean and error of the above rates denote the average values across all simulation platforms. Because the input (Poisson process of rate $47.3\,\mathrm{s}^{-1}$) and the connectivity (S randomly drawn synapses between M inputs and N neurons) is re-drawn for every simulation, the spiking activity varies and we average the execution time across 100 trials. For simulations using multiple threads, for each synapse model we consider the number of threads with the smallest execution time. The execution time is defined as the time to re-write synaptic weights, run the network and read out all spike times excluding the initialization of the simulation platform and building the network. The errors for energies E are likely to be large because of imprecise power measurements.

tivity of $R = 19.1\,\mathrm{Hz}$. With a power consumption of $P = 1.14\,\mathrm{MW}$ (Hennecke et al., 2012) we obtain $E = (P \cdot T)/(S \cdot R) = 83\,\mu\mathrm{J}$ for each synaptic transmission. For a different network simulated with different software (Gewaltig & Diesmann, 2007) on a different machine the efficiency is even worse. The *K computer* consumes $P = 12.66\,\mathrm{MW}$ of power (Uno, 2011), and 1 s of a network with $S = 10.4$ trillion synapses and an average firing rate of approximately $R = 10\,\mathrm{Hz}$ (Eppler, 2014) is simulated in $T = 40\,\mathrm{min}$ (Diesmann, 2013). This results in $E = 292\,\mathrm{mJ}$ for each synaptic transmission. Note that the worse performance of the latter compared to the first example is caused by different network models and simulation techniques. In comparison with the classifier network, the first example is on level with the desktop machine, but less energy-efficient than the *Spikey* system.

The above classifier benchmark does not include STDP, as it is the case for both networks in the supercomputer scenario. In case of the *Spikey* system, the energy efficiency does not increase for networks using STDP, because the acceleration is fixed and the power consumption of hardware emulations stays approximately constant. In contrast, the efficiency of simulations on desktop computers will decrease, if static synapses are replaced by synapses comprising STDP, because the runtime of simulations usually increases with model complexity.

Note that while prototyping networks, or using the *Spikey* system as a tool to interactively emulate networks (see all publications in this work), the workload on the host computer is not negligible (see also Section 3.3). Consequently, to be fair, the power consumption of the host computer has to be added to that of the *Spikey* system. However, image the *Spikey* chip to be embedded into a system of sensors and actuators that are also highly accelerated and spike-based. Then, after configuring the chip once, any further non-spiking communication to the outside world (including the host computer) is obsolete.

For biological brains, Attwell & Laughlin (2001) have reported an energy consumption of approximately 1 fJ for single synaptic transmissions (we assumed 12 kilo calories for each mole of ATP, see Alberts et al., 2007). Summarized, the neuromorphic chip *Spikey* represents a power efficient alternative to conventional computers, but is still far away from biological performance. Future silicon systems try to close this gap by, e.g., novel components, called memristors (Jo et al., 2010), or three-dimensional integration techniques (Koyanagi et al., 2001).

3.5 Hardware plasticity and learning

The *Spikey* neuromorphic system facilitates two types of plasticity: on-chip STDP and plasticity with the chip in the loop. On-chip STDP is implemented locally in each synapse, and hence allows for massively parallel learning, exploiting the full acceleration of the *Spikey* chip (see Publication III). However, the area efficient and hybrid implementation of STDP restricts the configurability of STDP: Due to its analog implementation, the time-dependence of hardware STDP (Publication II) is limited to exponential functions without offset (Schmidt, 2013, and Publication II and III), making it difficult to implement models with other curve shapes (e.g., Vogels et al., 2011; Nessler et al., 2013). The limited resolution and linear discretization of synaptic weights may be sufficient for additive and slightly multiplicative weight dependencies (Publication II), but are inappropriate to reflect strongly multiplicative and power law rules. However, the weight resolution can be virtually increased by feeding the same input to a second line of static synapses adding an offset to the plastic weights.

On network level, the hardware implementation and configuration of the time- and weight-dependency is sufficient for fine-grained synchrony detection, but the long integra-

tion of spike correlations, threshold effects, and common resets during the evaluation of correlations is not (Publication II). In feed-forward networks with random and slightly correlated input, due to the unidirectional connections, causal ($\Delta t > 0$) correlations between pre- and postsynaptic spikes will dominate anti-causal ($\Delta t < 0$) ones. Statistical fluctuations on short time-scales can break this domination, but are suppressed by the rather long integration of correlations caused by the low frequency with which synapses are processed and updated by the global controller (imagine common resets in Figure 5A of Publication II). Consequently, all synaptic weights tend to saturate at the maximum weight value on hardware. However, for highly correlated and structured input, the balance between potentiation and depression can be established by strong anti-causal correlations (Publication III).

In contrast to the above autonomous and on-chip plasticity, we demonstrated supervised learning on the *Spikey* system (Publication IV and liquid state machine Publication I). Networks are consecutively emulated on the chip, and the connectivity of these networks is optimized for pattern classification. Because learning is performed step-wise, using the chip in the loop exploits its acceleration and allows for rapid learning and classification. However, the costly pre- and post-processing of stimuli and chip configurations reduces the effective acceleration of the system (for reasons and possible improvements see Section 3.3).

While for a similar liquid state machine than in Publication I data sets for classification are linearly translated into firing rates (Probst, 2011), in the olfaction-inspired classifier, the placement of virtual receptors is rather complex and computationally expensive (Publication IV). In future systems, on-chip plasticity can be used to place these receptors, the classification of which can then be learned by using the chip in the loop.

Other accelerated systems also implement plastic synapses (Wijekoon & Dudek, 2012), but functional networks have only be shown for real time systems (e.g., Bofill-i Petit & Murray, 2004; Cameron et al., 2005; Hafliger, 2007; Mitra et al., 2009; Seo et al., 2011; Davies et al., 2012; Diehl & Cook, 2014). Moreover, we do not know of any analog neuromorphic system with a comparable number of plastic synapses. Even for digital systems, STDP is difficult to implement, because both pre- and postsynaptic activity has to be available at the synapse (Diehl & Cook, 2014). To reduce the computational complexity of STDP, Davies et al. (2012) have suggested to replace the purely spike-based STDP rule by a forecast rule involving the membrane potential.

Although, at the moment, hardware STDP is limited to few network models because of the common resets, adding a second reset line for future versions of the chip would extend its applicability substantially (for details, see Publication II). Then, hardware networks with STDP can be used to detect small correlations, which is the underlying functionality for many network models (e.g., Bofill-i Petit & Murray, 2004). To control on-chip chip learning, e.g., to turn plasticity on and off, the on-chip update controller could be extended with mechanisms that would allow to regulate learning rates or to mimic, e.g., neuromodulators. For the wafer-scale system, a dedicated on-chip microprocessor is under development to fulfill this task (Friedmann et al., 2013; Friedmann, 2013), also enabling quite complex tasks like expectation maximization (for details, seeSection 3.7).

3.6 Robustness of neuromorphic implementations

The robustness of data processing is a key feature of neural networks. By robustness we do not only mean the stability of the network and its convergence to a solution (see, e.g., Figure 3A in Publication IV and Figure 4D, 6C, 7F in Publication I), but also its

compensation for temporal and spatial variations of the computing substrate, or loss of single components, i.e., neurons and synapses (see Publication V, Figure 8 in Publication II and Petrovici et al., 2014). Without any dedicated calibrations of network parameters, the coefficient of variation of membrane time constants on *Spikey* is approximately 40% (see Figure 3 in Publication I), excitatory postsynaptic potentials vary by approximately 50% (see Figure 3 in Publication III) and the firing rates of neurons with constant current influx vary by roughly 50% (see Figure 4 in Publication V). Note that the variation of neuron and synapse parameters depends on their mean values and on the operation point of the network (for further listings of variations, see Brüderle et al., 2009).

In order to test the robustness of network models against parameter variations inherent to the hardware, we randomized the mapping from the model description to the hardware resources. For different realizations of the same network model, each neuron and synapse in the model description was randomly mapped to hardware representations of these. Because neuron and synapse parameters vary in hardware due to fixed-pattern noise, this procedure assigns different parameter sets to the model description. The average network performance across these realizations can then be used as a criterion for the robustness of the neuromorphic implementation of the network model (see Publication III and V).

Further, we tested the robustness of network models by varying the stimulation of these between hardware emulations. If the stimulation is specified by a stochastic process, spikes were re-drawn between emulations (Publication IV and winner-take-all model in Publication I). Otherwise, spike times were jittered (Publication III and liquid state machine in Publication I). In future studies, randomization could also be applied to the original data set, e.g., images may be translated, scaled or rotated, which may not only put the robustness of the network to the test, but may even improve classification performance (Cireşan et al., 2012).

Compared to variations induced by fixed-pattern noise in the computing substrate and the stochasticity in the input, trial-to-trial variations caused by temporal noise are usually small (see Publication V). Nevertheless, the robustness against trail-to-trial variability can be improved by elongating network emulations or averaging across multiple and identical repetitions of these. Rapid repetitions of trials are facilitated by the fact that the input can be buffered in the on-chip memory, and hence the stimulus has to be encoded and transferred to the system only once.

In this work, we developed three methods to improve the robustness of network models against fixed-pattern noise. First, fixed-pattern noise is compensated by calibrating neuron and synapse parameters, as used in Publication I, II, IV and V. Second, diversities in the response properties of neurons and synapses are reduced by averaging across populations of these, respectively. For example, a larger group size and sparser connectivity support the stability of signal propagation in synfire chains (Publication I), and improve the precision of encoding and processing data in the olfaction-inspired classifier (Publication IV). Third, even without prior calibration, the network can be designed to compensate for these variations autonomously, e.g., by using on-chip STDP to select these synapses that have the strongest simultaneous impact on the postsynaptic neuron (Publication III). Thereby, we obtain a signal more precise than the size of the membrane time constant, although the neuron receives noisy input via heterogeneous synapses. Such networks can be seen as self-calibrating by their design and are especially useful if prior calibration is too complex or the network has to adapt to a changing environment, like temperature or supply voltage fluctuations that are abundant in electronic devices.

Naturally, each network model has its limitation in terms of robustness against fixed-pattern noise. While some networks are robust and perform well on the uncalibrated

system (see Publication III and V), other networks rely on adequate calibration (see Publication IV and cortical model as well as winner-take-all network in Publication I). If calibration is impossible, e.g., due to parameters that are shared between neurons and synapses, respectively, fixed-pattern noise may incorrigibly distort network performance (e.g., see asymmetry in STDP curves in Publication II).

In order to compensate for the variability of response properties of neurons, we propose homeostatic mechanisms, e.g., for the firing rates of neurons, to be implemented in future systems. This would facilitate, e.g., the speed and convergence of learning (Nessler et al., 2013) and the decorrelating effect by inhibitory feedback (Publication V).

Concluding, throughout this study, the extend of fixed-pattern noise was no hard limitation for the implementation of network models, but usually additional resources had to be spent for its compensation. For example, in case of the classifier network, the size of neuron populations was larger than necessary for homogeneous neurons and synapses, which in turn reduced the number of virtual receptors and data classes (Publication IV). Detailed investigations of the ideal ratio between network size and the size of fixed-pattern noise may be subject to future studies, and the further development of intrinsic plasticity mechanisms (like, e.g., Publication III and see homeostasis above) may shift this tradeoff towards the neuron count.

3.7 Conclusion and perspectives

Throughout this work we presented tools, methods, scientific approaches, and examples for the exploration of brain-inspired algorithms on analog neuromorphic hardware.

During the design phase of neuromorphic systems, a compromise between the model's level of detail and its cost in terms of chip area and power efficiency has to be made. To verify the functionality of the final implementation of the chip in the context of single components and on network level, we presented a preparative study on the example of hardware STDP (Publication II). In the course of this study we developed strategies and the infrastructure for the generic configuration of STDP with discretized weights. Additionally, the prediction of potential bottlenecks led to improvements in the design of the wafer-scale system (Friedmann et al., 2013; Friedmann, 2013). Concluding, we recommend this "good practice" of preparative studies, because costly and time-consuming revisions of chip designs can be avoided, and the experience gained with a prototype, even though in software, likely accelerates the implementation of algorithms.

The versatility of the *Spikey* system is demonstrated by the attached publications, and the system matured to a valuable prototyping platform for neural algorithms. The *Spikey* system was applied to solve real-world problems (Publication I and IV), and to deepen the understanding of dynamics in recurrent networks on heterogeneous substrates (Publication V). Additionally, we showed efficient implementations of established network models with both static and plastic synapses (Publication I and III).

However, during the implementation of these models on hardware, the characteristics of the hardware had to be considered. For example, network models had to be reduced in size, had to be adapted to differing neuron and synapse models, and had to be modified to improve their robustness against fixed-pattern and temporal noise (see, e.g., networks in Publication I). For most networks, this adaptation resulted in less network modules and worse network performance. For the presented networks, the LIF neuron with synapses comprising short-term plasticity turned out to be sufficient to replace more complex neuron models (see cortical model in Publication I and Petrovici et al., 2014, for a similar approach). Up to now it is unclear to what extend networks of LIF neurons can replace

networks comprising dendritic computation or neurons with multiple compartments.

Fixed-pattern noise, inevitable in analog neuromorphic hardware, was compensated by calibration methods, population coding, and plasticity (for details, see Section 3.6). In contrast, temporal noise did not critically distort the performance of network emulations, and could easily be reduced by averaging over multiple trials. For many network models, the precision of spike times is not necessary, but the relative timing between spikes is (e.g., see Publication III and V). Hence, usually the network performance does not suffer from spike jitter, as long as spike times are consistent within the network. In addition, some studies require rather precise repeatability of network emulations, e.g., to record observables consecutively, which is given for most networks, especially if their structure is dominated by feed-forward connections (see Figure 3 in Publication V). An exception are networks with chaotic dynamics, e.g., highly recurrent networks (Publication V), in which the trajectory of network activity is highly sensitive to small disturbances of single variables, e.g., jittered spike times. In that case and when emulations should precisely match theoretical models, digital neuromorphic systems should be preferred to analog systems, because digital systems are less noisy, or not noisy at all (Merolla et al., 2014; Furber et al., 2014). If network models require precision beyond methodological requirements, this could indicate that they are biologically less plausible, because biological networks are noisy, too (e.g., Mainen & Sejnowski, 1995; Shafi et al., 2007).

Most network models shown in Publication I had to be shrinked to fit on the *Spikey* system, because the neuron count on *Spikey* is rather small. In order to tackle more complex problems requiring larger networks, networks have to be modularized, if possible, or otherwise larger systems have to be used (see wafer-scale system as discussed below). Through modularization of the network, the activity of single modules can be buffered, and consecutively fed into depending modules that are implemented on the same chip after re-configuring (e.g., see Kriener, 2014). In future studies, this approach may also be applied to the presented olfaction-inspired classifier network. The decorrelation and association layer could be emulated in consecutive emulations, and hence more neurons are available for both layers allowing for an increase of the number of glomeruli and data labels. However, modularization slows down the execution of networks through buffering network activity and re-configuring the chip, and are therefore only an intermediate step towards neuromorphic implementations of the full network.

In this work, we have shown that the *Spikey* chip and system provide considerable computational speed, which is especially important for investigations of network models comprising long-time plasticity, both on-chip and with the chip in the loop (Section 3.5), extensive parameter sweeps, or statistical analysis (Section 3.3). The latter is required anyway if effects on the performance by either fixed-pattern or temporal noise should be averaged out (see Section 3.6). In addition, fast responses to stimuli may be useful for practical applications, like decision making or object recognition (see Publication IV).

In the scope of this thesis, the performance of the system could be significantly improved compared to former work (Brüderle, 2009). In particular, we improved the software infrastructure, including the interface between the *Spikey* hardware and the standardized network description language `PyNN`, and commissioned the USB version of the *Spikey* system. Nevertheless, there is still a great potential to further improve the performance of the system (for details see Section 3.3).

To further develop the classifier network presented in Publication IV, the placement of virtual receptors in data space could be performed by stochastic network models. Büsing et al. (2011) propose that neural activity represents samples from underlying probability distributions and networks of spiking neurons emulate powerful algorithms for reasoning,

i.e., carry out probabilistic inference. To run these algorithms on the *Spikey* chip, its deterministic spiking neurons have to be driven into an activity regime, in which they achieve the correct firing statistics to sample from a well-defined target distribution (Petrovici et al., 2013; Probst et al., 2014). The performance of sampling, however, depends on the background stimulation that keeps these neurons in this activity regime. In order to use the existing hardware and to reduce the costly generation of appropriate background activity and its transfer to the hardware system, this activity could be provided by recurrent networks that are implemented on the same computing substrate (see Publication V). The effect of different statistics of this background activity is left for further studies. Through STDP sampling networks are shown to perform unsupervised classification on high-dimensional spike inputs by performing *expectation maximization* (Nessler et al., 2009, 2013). In the *Spikey* system, the configurability of STDP is rather limited, and neurons miss homeostatic mechanisms, which makes it difficult to implement this network model (see Section 1.5.2). However, the wafer-scale system may offer plasticity features (Friedmann et al., 2013; Friedmann, 2013) to overcome these obstacles. This would enable expectation maximization in stochastic networks of spiking neurons exploiting accelerated neuromorphic hardware (work in progress by Breitwieser, 2014, among others).

In its current state, the *Spikey* system is ready to be applied to real-world computing problems, like object recognition, decision making and data mining (see Publication IV). However, performance in terms of network size and power efficiency can be further increased by customizing the previously configurable *Spikey* chip to a less configurable version, which is dedicated to a specific task. Furthermore, neuromorphic circuits and CPUs could be densely interconnected on a single chip to integrate the advantages of both computing architectures (see also Section 1.5.3).

Although the *Spikey* chip has hardly more computational power than a modern desktop computer (i.a., see Table 3.1), the integration of many neuromorphic chips on a wafer allows to scale this technology to network sizes that may allow for performance and efficiency not reachable by conventional clusters of computers (Schemmel et al., 2010; Brüderle et al., 2011; Müller, 2014). Highly accelerated and massively parallel emulations of large networks enable for rapid exploration of brain-inspired algorithms, especially effective for networks requiring long runtimes, e.g., long-term learning experiments, or intensive parameter sweeps. Beyond the utilization shown in this work, such systems could be used to evolve neural networks by genetic algorithms (work in progress by Lackner, 2014). These algorithms could make manual configuration obsolete, and have the potential to compensate or even exploit variations of the hardware system. Usually, the differences between network configurations are sparse during evolution, and hence differential configuration may allow for a high throughput of genes. In cases, where automated tuning of network parameters is not feasible, graphical user interfaces may be used for this purpose, taking advantage of the rapid response of accelerated hardware.

This work is not restricted only to the *Spikey* system, but can also be seen as a preparative study for the wafer-scale system, because the methodology to develop and implement network models are likely to be transferable between these systems. Besides calibration techniques and configuration strategies (Publication I and II), networks models established on *Spikey* are likely to run on the wafer-scale system, because comparable configuration space, fixed-pattern noise and temporal noise is expected. Note, however, that the wafer-scale system consists of neuromorphic chips that are dedicated for wafer-scale integration and comprise different neuron and synapse circuits than the *Spikey* chip.

In case of the end of Moore's law (Moore, 2006; ITRS, 2013), postulating a long-term trend of the transistor count in dense integrated circuits (and hence of the computational

power), von-Neumann-like architectures may be replaced or complemented by novel hardware architectures. The neuromorphic approach proposes a fundamentally different utilization of silicon resources and is hence a promising candidate to increase the computational power and efficiency beyond that of conventional computers (including graphics processing units (GPUs) and many-core CPUs and systems, e.g., Intel (2014) and Parallella (2014)). Once neural features and algorithms are sufficiently identified and neuromorphic devices mature to a state where mass production is feasible, such devices may shape the dimension, performance, and power efficiency of technical devices, like, e.g., mobile phones. Consequently, research on the implementation of brain-inspired algorithms may not only be beneficial for understanding brains, but may also facilitate the daily applicability of neuromorphic hardware systems.

The presented publications demonstrate that the neuromorphic *Spikey* system matured from a laboratory-only system to an ecosystem of neuromorphic hardware and software that supports practical applications and helps to answer neuroscientific questions.

Bibliography

Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nat. Neurosci. 3*(Supplement), 1178–1183.

Ahrens, M. B., Orger, M. B., Robson, D. N., Li, J. M., & Keller, P. J. (2013). Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nat. Meth. 10*(5), 413–420.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., & Walter, P. (2007). *Molecular Biology of the Cell* (5 ed.). Garland Science.

Ananthanarayanan, R., Esser, S. K., Simon, H. D., & Modha, D. S. (2009). The cat is out of the bag: Cortical simulations with 10^9 neurons and 10^{13} synapses. In *Supercomputing 09: Proceedings of the ACM/IEEE SC2009 Conference on High Performance Networking and Computing*, Portland, OR.

Arthur, J., Merolla, P., Akopyan, F., Alvarez, R., Cassidy, A., Chandra, S., Esser, S., Imam, N., Risk, W., Rubin, D., Manohar, R., & Modha, D. (2012). Building block of a programmable neuromorphic substrate: A digital neurosynaptic core. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–8.

arXiv.org (2014). arxiv.org. `http://www.arxiv.org/`. Online; accessed December 2, 2014.

Attwell, D., & Laughlin, S. B. (2001). An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow & Metabolism 21*, 1133–1145.

Badoni, D., Giulioni, M., Dante, V., & Del Giudice, P. (2006). An aVLSI recurrent network of spiking neurons with reconfigurable and plastic synapses. In *Proceedings of the 2006 International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, pp. 4. IEEE Press.

Benjamin, B., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A., Bussat, J., Alvarez-Icaza, R., Arthur, J., Merolla, P., & Boahen, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE 102*(5), 699–716.

Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci. 18*, 10464–10472.

Bi, G., & Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annu. Rev. Neurosci. 24*, 139–66.

Bill, J. (2008). Self-stabilizing network architectures on a neuromorphic hardware system. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 08-44.

Bill, J., Schuch, K., Brüderle, D., Schemmel, J., Maass, W., & Meier, K. (2010). Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity. *Front. Comput. Neurosci. 4*(129).

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Bofill-i Petit, A., & Murray, A. (2004). Synchrony detection and amplification by silicon neurons with STDP synapses. *IEEE Trans. Neural Netw. 15*(5), 1296–1304.

Bontorin, G., Renaud, S., Garenne, A., Alvado, L., Le Masson, G., & Tomas, J. (2007). A real-time closed-loop setup for hybrid neural networks. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 3004–3007.

Brain Corporation (2014). Company website. `http://www.braincorporation.com`. Online; accessed November 7, 2014.

Brain/MINDS (2014). Brain Mapping by Integrated Neurotechnologies for Disease Studies. `http://brainminds.jp/en/`. Online; accessed November 13, 2014.

Brainnetome (2014). of memory impairment in schizophrenia, national natural science foundation of china, 2012-2015. `http://www.brainnetome.org/brainnetomeproject/project`. Online; accessed November 13, 2014.

BrainScaleS (2014). Project website. Available at: `http://www.brainscales.eu`.

Breitwieser, O. (2014). Personal communication.

Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., & Degnan, B. (2013). A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Trans. Biomed. Circuits Syst. 7*(1), 71–81.

Brüderle, D. (2009). *Neuroscientific Modeling with a Mixed-Signal VLSI Hardware System*. Doctoral thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 09-30.

Brüderle, D. (2014). Personal communication.

Brüderle, D., Bill, J., Kaplan, B., Kremkow, J., Meier, K., Müller, E., & Schemmel, J. (2010). Simulator-like exploration of cortical network architectures with a mixed-signal VLSI system. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, France. IEEE Press.

Brüderle, D., Müller, E., Davison, A., Muller, E., Schemmel, J., & Meier, K. (2009). Establishing a novel modeling tool: A Python-based interface for a neuromorphic hardware system. *Front. Neuroinformatics 3*(17).

Brüderle, D., Petrovici, M., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., Grübl, A., Wendt, K., Müller, E., Schwartz, M.-O., Husmann de Oliveira, D., Jeltsch, S., Fieres, J., Schilling, M., Müller, P., Breitwieser, O., Petkov, V., Muller, L., Davison, A. P., Krishnamurthy, P., Kremkow, J., Lundqvist, M., Muller, E., Partzsch, J., Scholze, S., Zühl, L., Destexhe, A., Diesmann, M., Potjans, T. C., Lansner, A., Schüffny, R., Schemmel, J., & Meier, K. (2011). A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol. Cybern. 104*, 263–296.

Büsing, L., Bill, J., Nessler, B., & Maass, W. (2011). Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput. Biol. 7*(11), e1002211.

Cameron, K., Boonsobhak, V., Murray, A., & Renshaw, D. (2005). Spike timing dependent plasticity (STDP) can ameliorate process variations in neuromorphic VLSI. *IEEE Trans. Neural Netw. 16*(6), 1626–1637.

Cassenaer, S., & Laurent, G. (2007). Hebbian STDP in mushroom bodies facilitates the synchronous flow of olfactory information in locusts. *Nature 448*(7154), 709–713.

Cassenaer, S., & Laurent, G. (2012). Conditional modulation of spike-timing-dependent plasticity for olfactory learning. *Nature 482*(7383), 47–52.

Cireşan, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks 32*, 333–338.

Clopath, C., Büsing, L., Vasilaki, E., & Gerstner, W. (2010). Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nat. Neurosci. 13*, 344–352.

Connors, B. W., & Gutnick, M. J. (1990). Intrinsic firing patterns of diverse neocortical neurons. *TINS 13*(3), 99–104.

Coomans, W., Gelens, L., Beri, S., Danckaert, J., & Van der Sande, G. (2011). Solitary and coupled semiconductor ring lasers as optical spiking neurons. *Phys. Rev. E 84*, 036209.

Davies, S., Galluppi, F., Rast, A., & Furber, S. (2012). A forecast-based STDP rule suitable for neuromorphic implementation. *Neural Networks 32*(0), 3–14.

Davison, A., Brüderle, D., Eppler, J., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., & Yger, P. (2008). PyNN: a common interface for neuronal network simulators. *Front. Neuroinformatics 2*(11). doi:10.3389/neuro.11.011.2008.

Dayan, P., & Abbott, L. F. (2001). *Theoretical Neuroscience*. Cambridge: MIT Press.

Diehl, P., & Cook, M. (2014). Efficient implementation of STDP rules on SpiNNaker neuromorphic hardware. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pp. 4288–4295.

Diesmann, M. (2013). 10.4 trillion synapses simulated on supercomputer. `http://www.jara.org/en/research/jara-brain/news/details/2013/104-trillion-synapses-simulated-on-supercomputer/`. Online; accessed November 17, 2014.

Dodt, H.-U., Leischner, U., Schierloh, A., Jahrling, N., Mauch, C. P., Deininger, K., Deussing, J. M., Eder, M., Zieglgansberger, W., & Becker, K. (2007). Ultramicroscopy: three-dimensional visualization of neuronal networks in the whole mouse brain. *Nat. Meth. 4*(4), 331–336.

Eppler, J. (2014). Personal communication.

FACETS (2010). Fast Analog Computing with Emergent Transient States, project website. Available at: `http://www.facets-project.org`.

Floreano, D., & Mattiussi, C. (2001). Evolution of spiking neural controllers for autonomous vision-based robots. In *Evolutionary Robotics. From Intelligent Robotics to Artificial Life*, pp. 38–61. Springer.

Friedmann, S. (2009). Extending a hardware neural network beyond chip boundaries. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 09-41.

Friedmann, S. (2013). *A new approach to learning in neuromorphic hardware*. Doctoral thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 13-86.

Friedmann, S., Frémaux, N., Schemmel, J., Gerstner, W., & Meier, K. (2013). Reward-based learning under hardware constraints - using a RISC processor embedded in a neuromorphic substrate. *Front. Neurosci. 7*(160).

Froemke, R. C., Merzenich, M. M., & Schreiner, C. E. (2007). A synaptic memory trace for cortical receptive field plasticity. *Nature 450*(7168), 425–429.

Furber, S., Galluppi, F., Temple, S., & Plana, L. (2014). The SpiNNaker Project. *Proc. IEEE 102*(5), 652–665.

Galluppi, F., Rast, A., Davies, S., & Furber, S. (2010). A general-purpose model translation system for a universal neural chip. In K. Wong, B. Mendis, & A. Bouzerdoum (Eds.), *Neural Information Processing. Theory and Algorithms*, Volume 6443 of *Lecture Notes in Computer Science*, pp. 58–65. Springer Berlin Heidelberg.

Gerstner, W., Kempter, R., van Hemmen, J. L., & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature 383*, 76–78.

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia 2*(4), 1430.

Gibson, J. R., Beierlein, M., & Connors, B. W. (1999). Two networks of electrically coupled inhibitory neurons in neocortex. *Nature 402*, 75–79.

Giulioni, M., Camilleri, P., Dante, V., Badoni, D., Indiveri, G., Braun, J., & Del Giudice, P. (2008). A VLSI network of spiking neurons with plastic fully configurable "stop-learning" synapses. In *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, pp. 678–681.

Government of Canada (2014). Government of Canada reveals 32 new brain research projects. `http://news.gc.ca/web/article-en.do?mthd=index&crtr.page=1&nid=884079`. Online; accessed November 13, 2014.

Graupner, M., & Brunel, N. (2012). Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. *Proc. Natl. Acad. Sci. USA 109*(10), 3991–3996. doi: 10.1073/pnas.1109359109.

Gray, C., & McCormick, D. (1996). Chattering cells: superficial pyramidal neurons contributing to the generation of synchronous oscillations in the visual cortex. *Science 274*, 109–113.

Gross, G. W. (2011). Multielectrode arrays. *Scholarpedia 6*(3), 5749.

Grübl, A. (2007). *VLSI Implementation of a Spiking Neural Network*. Doctoral thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 07-10.

Grübl, A. (2014). Personal communication.

Hafliger, P. (2007). Adaptive WTA with an analog VLSI neuromorphic learning chip. *IEEE Trans. Neural Netw. 18*(2), 551–572.

Hartel, A. (2014). Personal communication.

Hasler, J., & Marr, H. B. (2013). Finding a roadmap to achieve large neuromorphic hardware systems. *Front. Neurosci. 7*(118). doi: 10.3389/fnins.2013.00118.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York: John Wiley & Sons.

Helmchen, Fritjofand Denk, W. (2005). Deep tissue two-photon microscopy. *Nat. Meth. 2*(12), 932–940.

Hennecke, M., Frings, W., Homberg, W., Zitz, A., Knobloch, M., & Böttiger, H. (2012). Measuring power consumption on IBM Blue Gene/P. *Computer Science - Research and Development 27*(4), 329–336.

Hines, M., & Carnevale, N. T. (1997). The NEURON simulation environment. *Neural Comput. 9*, 1179–1209.

Hines, M. L., Markram, H., & Schürmann, F. (2008). Fully implicit parallel simulation of single neurons. *J. Comput. Neurosci. 25*(3), 439–448.

Hock, M. (2014a). *Modern Semiconductor Technologies for Neuromorphic Hardware*. Doctoral thesis, Ruprecht-Karls-Universität Heidelberg. `http://archiv.ub.uni-heidelberg.de/volltextserver/17129/`.

Hock, M. (2014b). Personal communication.

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

HP (2010). MoNETA: A mind made from memristors. `http://spectrum.ieee.org/robotics/artificial-intelligence/moneta-a-mind-made-from-memristors`. Online; accessed November 7, 2014.

Human Brain Project (2014). Project website. Available at: `http://www.humanbrainproject.eu`.

Hurtado, A., Schires, K., Henning, I. D., & Adams, M. J. (2012). Investigation of vertical cavity surface emitting laser dynamics for neuromorphic photonic systems. *Appl. Phys. Lett. 100*(10), 103703.

IBM (2014). Brain power. `http://www.research.ibm.com/cognitive-computing/neurosynaptic-chips.shtml`. Online; accessed November 7, 2014.

Indiveri, G. (2001). A neuromorphic VLSI device for implementing 2D selective attention systems. *IEEE Trans. Neural Netw. 12*(6), 1455–1463.

Indiveri, G. (2012). Personal communication.

Indiveri, G., Chicca, E., & Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw. 17*(1), 211–221.

Indiveri, G., Chicca, E., & Douglas, R. (2009). Artificial cognitive systems: From VLSI networks of spiking neurons to neuromorphic cognition. *Cognitive Computation 1*(2), 119–127.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., & Boahen, K. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci. 5*(73).

Intel (2014). Intel® Xeon Phi™ product family: Product brief. `http://www.intel.com/content/www/us/en/high-performance-computing/high-performance-xeon-phi-coprocessor-brief.html`. Online; accessed December 1, 2014.

ITRS (2013). International technology roadmap for semiconductors. `http://www.itrs.net/Links/2013ITRS/Summary2013.htm`. Online; accessed November 13, 2014.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb. Cortex 17*(10), 2443–2452.

Jeltsch, S. (2010). Computing with transient states on a neuromorphic multi-chip environment. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 10-54.

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., & Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Letters 10*(4), 1297–1301.

Kaplan, B. (2008). Self-organization experiments for a neuromorphic hardware device. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 08-42.

Kempter, R., Gerstner, W., & van Hemmen, J. L. (2001). Intrinsic stabilization of output rates by spike-based Hebbian learning. *Neural Comput. 12*, 2709–2742.

Koyanagi, M., Nakagawa, Y., Lee, K.-W., Nakamura, T., Yamada, Y., Inamura, K., Park, K.-T., & Kurino, H. (2001). Neuromorphic vision chip fabricated using three-dimensional integration technology. In *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, pp. 270–271.

Kriener, L. (2014). Binaural sound localization on neuromorphic hardware. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 14-92.

Kumar, N. M., & Gilula, N. B. (1996). The gap junction communication channel. *Cell 84*(3), 381–388.

Kunkel, S., Potjans, T. C., Eppler, J. M., Plesser, H. E., Morrison, A., & Diesmann, M. (2012). Meeting the memory challenges of brain-scale simulation. *Front. Neuroinform. 5*, 35.

Kunkel, S., Schmidt, M., Eppler, J. M., Masumoto, G., Igarashi, J., Ishii, S., Fukai, T., Morrison, A., Diesmann, M., & Helias, M. (2014). Spiking network simulation code for petascale computers. *Frontiers in Neuroinformatics 8*(78).

Lackner, S. (2014). Personal communication.

London, M., & Häusser, M. (2005). Dendritic computation. *Annu. Rev. Neurosci. 28*, 503–532.

Maass, W. (1996). Lower bounds for the computational power of networks of spiking neurons. *Neural Comput. 8*(1), 1–40.

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks 10*(9), 1659–1671.

Mainen, Z. F., & Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science 268*, 1503–1506.

Marder, E., & Goaillard, J.-M. (2006). Variability, compensation and homeostasis in neuron and network function. *Nat. Rev. Neurosci. 7*(7), 563–574.

Markram, H., Toledo-Rodriguez, M., Wang, Y., Gupta, A., Silberberg, G., & Wu, C. (2004). Interneurons of the neocortical inhibitory system. *Nat. Rev. Neurosci. 5*(10), 793–807.

McCormick, D. A., Connors, B. W., Lighthall, J. W., & Prince, D. A. (1985). Comparative electrophysiology of pyramidal and sparsely spiny neurons of the neocortex. *J. Neurophysiol. 54*(4), 782–806.

Mead, C. (1990). Neuromorphic electronic systems. *Proc. IEEE 78*(10), 1629–1636.

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., & Modha, D. (2011). A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In *Proceedings of the 2011 Custom Integrated Circuits Conference (CICC)*, San Jose, USA, pp. 1–4. IEEE Press.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S. K., Appuswamy, R., Taba, B., Amir, A., Flickner, M. D., Risk, W. P., Manohar, R., & Modha, D. S. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science 345*(6197), 668–673.

Mitra, S., Fusi, S., & Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Circuits Syst. 3*(1), 32–42.

Moore, G. E. (2006). Cramming more components onto integrated circuits, reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *Solid-State Circuits Society Newsletter, IEEE 11*(5), 33–35.

Morrison, A., Aertsen, A., & Diesmann, M. (2007). Spike-timing dependent plasticity in balanced random networks. *Neural Comput. 19*, 1437–1467.

Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike-timing. *Biol. Cybern. 98*, 459–478.

Müller, E. (2008). Operation of an imperfect neuromorphic hardware device. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 08-43.

Müller, E. (2014). *Novel Operation Modes of Accelerated Neuromorphic Hardware*. Doctoral thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 14-98.

Müller, P. (2011). Distortions of neural network models induced by their emulation on neuromorphic hardware devices. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 11-172.

Neftci, E., Chicca, E., Indiveri, G., & Douglas, R. (2011). A systematic method for configuring VLSI networks of spiking neurons. *Neural Comput. 23*(10), 2457–2497.

Nessler, B., Pfeiffer, M., Buesing, L., & Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput. Biol. 9*(4), e1003037.

Nessler, B., Pfeiffer, M., & Maass, W. (2009). STDP enables spiking neurons to detect hidden causes of their inputs. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, & A. Culotta (Eds.), *Advances in Neural Information Processing Systems 22*, pp. 1357–1365. Curran Associates, Inc.

Nowak, L. G., Azouz, R., Sanchez-Vives, M. V., Gray, C. M., & McCormick, D. A. (2003). Electrophysiological classes of cat primary visual cortical neurons in vivo as revealed by quantitative analyses. *J Neurophysiol 89*(3), 1541–66.

Pakkenberg, B., Pelvig, D., Marner, L., Bundgaard, M. J., Gundersen, H. J. G., Nyengaard, J. R., & Regeur, L. (2003). Aging and the human neocortex. *Experimental Gerontology 38*(1–2), 95–99.

Parallella (2014). Parallella. `http://www.parallella.org/`. Online; accessed December 1, 2014.

Perin, R., Berger, T. K., & Markram, H. (2011). A synaptic organizing principle for cortical neuronal groups. *Proc. Natl. Acad. Sci. USA 108*(13), 5419–5424.

Petkov, V. (2012). Toward belief propagation on neuromorphic hardware. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 12-23.

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., & Meier, K. (2013). Stochastic inference with deterministic spiking neurons.

Petrovici, M. A., Vogginger, B., Müller, P., Breitwieser, O., Lundqvist, M., Muller, L., Ehrlich, M., Destexhe, A., Lansner, A., Schüffny, R., Schemmel, J., & Meier, K. (2014). Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PLoS ONE 9*(10), e108590.

Pfeil, T. (2011). Configuration strategies for neurons and synaptic learning in large-scale neuromorphic hardware systems. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 11-34.

Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., Schmuker, M., Brüderle, D., Schemmel, J., & Meier, K. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci. 7*(11).

Pfeil, T., Jordan, J., Tetzlaff, T., Grübl, A., Schemmel, J., Diesmann, M., & Meier, K. (2014). The effect of heterogeneity on decorrelation mechanisms in spiking neural networks: a neuromorphic-hardware study.

Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., & Meier, K. (2012). Is a 4-bit synaptic weight resolution enough? - Constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci. 6*(90).

Pfeil, T., Scherzer, A.-C., Schemmel, J., & Meier, K. (2013). Neuromorphic learning towards nano second precision. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–5. IEEE Press.

Pfister, J.-P., & Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *J. Neurosci. 26*, 9673–9682.

Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., & Delbruck, T. (2014). Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proc. IEEE 102*(10), 1470–1484.

Potjans, T. C., & Diesmann, M. (2014). The cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network model. *Cereb. Cortex 24*(3), 785–806. doi: 10.1093/cercor/bhs358.

Probst, D. (2011). Analysis of the liquid computing paradigm on a neuromorphic hardware system. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 11-47.

Probst, D., Petrovici, M. A., Bytschok, I., Bill, J., Pecevski, D., Schemmel, J., & Meier, K. (2014). Probabilistic inference in discrete spaces with networks of LIF neurons.

Qualcomm (2013). Introducing Qualcomm Zeroth Processors: Brain-inspired computing. `https://www.qualcomm.com/news/onq/2013/10/10/introducing-qualcomm-zeroth-processors-brain-inspired-computing`. Online; accessed November 7, 2014.

Renaud, S., Tomas, J., Lewis, N., Bornat, Y., Daouzli, A., Rudolph, M., Destexhe, A., & Saïghi, S. (2010). PAX: A mixed hardware/software simulation platform for spiking neural networks. *Neural Networks 23*(7), 905–916.

Rotter, S., & Diesmann, M. (1999). Exact digital simulation of time-invariant linear systems with applications to neuronal modeling. *Biol. Cybern. 81*(5/6), 381–402.

Roy, K., Sharad, M., Fan, D., & Yogendra, K. (2014). Computing with spin-transfer-torque devices: Prospects and perspectives. In *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, pp. 398–402.

Sakmann, B., & Neher, E. (1984). Patch clamp techniques for studying ionic channels in excitable membranes. *Annu. Rev. Physiol. 46*(1), 455–472.

Schemmel, J. (2014). Personal communication.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 International Symposium on Circuits and Systems (ISCAS)*, Paris, pp. 1947–1950. IEEE Press.

Schemmel, J., Brüderle, D., Meier, K., & Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated I&F neurons. In *Proceedings of the 2007 International Symposium on Circuits and Systems (ISCAS)*, New Orleans, pp. 3367–3370. IEEE Press.

Schemmel, J., Grübl, A., Meier, K., & Müller, E. (2006). Implementing synaptic plasticity in a VLSI spiking neural network model. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, pp. 1–6. IEEE Press.

Scherzer, A.-C. (2013). Phase-locking on neuromorphic hardware. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 10-43.

Schilling, M. (2010). A highly efficient transport layer for the connection of neuromorphic hardware systems. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 10-09.

Schmidt, O. (2013). Characteristics of STDP on the spikey neuromorphic hardware system. Bachelor thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 13-108.

Schmuker, M. (2013). Personal communication.

Schmuker, M., Pfeil, T., & Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proc. Natl. Acad. Sci. USA 111*(6), 2081–2086.

SenseMaker (2005). Project website. Available at: `http://cordis.europa.eu/project/rcn/62746_en.html`.

Seo, J., Brezzo, B., Liu, Y., Parker, B., Esser, S., Montoye, R., Rajendran, B., Tierno, J., Chang, L., Modha, D., & Friedman, D. (2011). A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pp. 1–4.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gomez-Rodriguez, F., Camunas-Mesa, L., Berner, R., Rivas-Perez, M., Delbruck, T., Liu, S.-C., Douglas, R., Hafliger, P., Jimenez-Moreno, G., Ballcels, A., Serrano-Gotarredona, T., Acosta-Jimenez, A., & Linares-Barranco, B. (2009). CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory - processing - learning - actuating system for high-speed visual object recognition and tracking. *IEEE Trans. Neural Netw. 20*(9), 1417–1438.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodríguez, F., Kolle Riis, H., Delbrück, T., Liu, S.-C., Zahnd, S., Whatley, A. M., Douglas, R., Häfliger, P., Jimenez-Moreno, G., Civit, A., Serrano-Gotarredona, T., Acosta-Jiménez, A., & Linares-Barranco, B. (2006). AER building blocks for multi-layer multi-chip neuromorphic vision systems. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *anips*, pp. 1217–1224. Cambridge, MA: MIT Press.

Shafi, M., Zhou, Y., Quintana, J., Chow, C., Fuster, J., & Bodner, M. (2007). Variability in neuronal activity in primate cortex during working memory tasks. *Neuroscience 146*(3), 1082–1108.

Sharp, T., Galluppi, F., Rast, A., & Furber, S. (2012). Power-efficient simulation of detailed cortical microcircuits on SpiNNaker. *J. Neurosci. Methods 210*(1), 110–118.

Shastri, B. J., Tait, A. N., Nahmias, M. A., & Prucnal, P. R. (2014). Photonic spike processing: ultrafast laser neurons and an integrated photonic network. *arXiv preprint arXiv:1407.2917*.

Sjöström, J., & Gerstner, W. (2010). Spike-timing dependent plasticity. *Scholarpedia 5*(2), 1362.

Sjöström, P., Turrigiano, G., & Nelson, S. (2001). Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron 32*, 1149–1164.

Sjöström, P. J., Rancz, E. A., Roth, A., & Häusser, M. (2008). Dendritic excitability and synaptic plasticity. *Physiol. Rev.*, 769–840. doi:10.1152/physrev.00016.2007.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci. 3*(9), 919–926.

Stefanini, F. (2012). Hydroneuron. `https://www.youtube.com/watch?v=yNt9sYhasnI`. Online; accessed October 15, 2014.

Stein, R. B., Gossen, E. R., & Jones, K. E. (2005). Neuronal variability: Noise or part of the signal? *Nat. Rev. Neurosci. 6*, 389–397.

SyNAPSE (2014). Systems of Neuromorphic Adaptive Plastic Scalable Electronics, project website. Available at: `http://www.darpa.mil/Our_Work/DSO/Programs/Systems_of_Neuromorphic_Adaptive_Plastic_Scalable_Electronics_(SYNAPSE).aspx`.

Tait, A. N., Nahmias, M. A., Tian, Y., Shastri, B. J., & Prucnal, P. R. (2014). Photonic neuromorphic signal processing and computing. In M. Naruse (Ed.), *Nanophotonic Information Physics*, Nano-Optics and Nanophotonics, pp. 183–222. Berlin, Heidelberg: Springer.

The Brain Initiative (2014). Project website. Available at: `http://www.braininitiative.nih.gov/index.htm`.

Tsodyks, M., & Wu, S. (2013). Short-term synaptic plasticity. *Scholarpedia 8*(10), 3153.

Tsodyks, M. V., & Markram, H. (1997). The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proc. Natl. Acad. Sci. USA 94*, 719–723.

Uno, A. (2011). The K computer system overview. `http://www.fujitsu.com/downloads/TC/sc11/k-computer-system-overview-sc11.pdf`. Online; accessed November 17, 2014.

Vogels, T., Sprekeler, H., Zenke, F., Clopath, C., & Gerstner, W. (2011). Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science 334*(6062), 1569–1573.

Vogelstein, R. J., Mallik, U., Culurciello, E., Cauwenberghs, G., & Etienne-Cummings, R. (2007). A multichip neuromorphic system for spike-based visual information processing. *Neural Comput. 19*(9), 2281–2300.

Vogginger, B. (2010). Testing the operation workflow of a neuromorphic hardware system with a functionally accurate model. Diploma thesis, Ruprecht-Karls-Universität Heidelberg. HD-KIP 10-12.

von Neumann, J. (1993). First draft of a report on the EDVAC. *Annals of the History of Computing, IEEE 15*(4), 27–75. doi: 10.1109/85.238389.

Weckstrom, M. (2010). Intracellular recording. *Scholarpedia 5*(8), 2224.

Wijekoon, J. H., & Dudek, P. (2008). Compact silicon neuron circuit with spiking and bursting behaviour. *Neural Networks 21*(2–3), 524–534.

Wijekoon, J. H., & Dudek, P. (2012). VLSI circuits implementing computational models of neocortical circuits. *J. Neurosci. Methods 210*(1), 93–109.

www.thethirdsource.org, W. (2014). Structure of a neuron. `http://www.thethirdsource.org/wp-content/uploads/neuron.gif`. Online; accessed October 24, 2014.

Yu, T., & Cauwenberghs, G. (2010). Analog VLSI biophysical neurons and synapses with programmable membrane channel kinetics. *IEEE Trans. Biomed. Circuits Syst. 4*(3), 139–148.

Chapter 4

Appendix

4.1 Classifier benchmark

For benchmarking software simulations we used a desktop computer with an Intel® Core™ i5-4570 CPU @ 3.20GHz processor consuming $P = 84$ watt of power, and `NEST` version 2.2.2 with `PyNN` version 0.7.5.

For hardware emulations we used the *Spikey* system (carrier board version 1) with chip number 605 connected to the computer specified above. We used vmodule (db5bc413), SpikeyHAL (00755bc1) and pynn-hardware (c76e1a48) in their version specified by the commit hashs.

4.2 Chip usage

For Publication I, II and III we used station 111 (with chip number 444) and 315 (with chip number 445) on the backplane system 1. For Publication IV we used station 309 (with chip number 443) on the backplane system 3 and the USB system (carrier board version 1) with chip number 666. For Publication V we used the USB system (carrier board version 1) with chips number 603, 605 and 666.

Acknowledgments

Im Folgenden möchte ich mich bedanken bei

- meinem Erstgutachter und Mentor Professor Karlheinz Meier. Vielen Dank für ihr Vertrauen in mich, und dass ich an ihrer Vision teilhaben durfte.
- Professor Abigail Morrison, die sich als Zweitgutachterin zur Verfügung gestellt hat.
- den externen Koautoren der hier aufgeführten Publikationen. Insbesondere danke ich Michael, Jakob, und Tom für die besonders intensive und bereichernde Zusammenarbeit.
- Christoph, Sebastian S., Simon, Matthias und Paul für das Korrekturlesen.
- den Visionären, welche mich tatkräftig bei meinen Vorhaben unterstützt haben. Danke Christoph und Sebastian J. für die gemeinsame Zeit im Büro und eure Hilfe im Kampf mit den Computern. Danke Eric, Andi H., Paul und Matthias für die exzellente technische Unterstützung. Danke Mihai und Sebastian S. für fruchtbare Diskussionen. Danke Andi G., Johannes und Daniel (und andere) für die Vorarbeit bezüglich des *Spikey* Systems.
- meinen fleißigen Bachelorstudenten Sebastian L., Anki, Ole und Laura.
- meinen Freunden (obige Personen sind nicht ausgeschloßen) für die abwechslungsreiche Freizeitgestaltung und schöne Zeiten abseits der Wissenschaft.
- meiner Familie.
- meiner Liebe Eva samt Rosinchen.