Demonstrating Hybrid Learning in a Flexible Neuromorphic Hardware System

Simon Friedmann, Johannes Schemmel, *Member, IEEE*, Andreas Grübl, Andreas Hartel, Matthias Hock, and Karlheinz Meier

***Abstract*—We present results from a new approach to learning and plasticity in neuromorphic hardware systems: to enable flexibility in implementable learning mechanisms while keeping high efficiency associated with neuromorphic implementations, we combine a general-purpose processor with full-custom analog elements. This processor is operating in parallel with a fully parallel neuromorphic system consisting of an array of synapses connected to analog, continuous time neuron circuits. Novel analog correlation sensor circuits process spike events for each synapse in parallel and in real-time. The processor uses this pre-processing to compute new weights possibly using additional information following its program. Therefore, to a certain extent, learning rules can be defined in software giving a large degree of flexibility. Synapses realize correlation detection geared towards Spike-Timing Dependent Plasticity (STDP) as central computational primitive in the analog domain. Operating at a speed-up factor of 1000 compared to biological time-scale, we measure time-constants from tens to hundreds of micro-seconds. We analyze variability across multiple chips and demonstrate learning using a multiplicative STDP rule. We conclude that the presented approach will enable flexible and efficient learning as a platform for neuroscientific research and technological applications.**

***Index Terms*—Digital signal processing, learning, neuromorphic hardware, spike-time dependent plasticity, synapse circuit.**

I. Introduction

In THE modern landscape of information technology machine learning is gaining more and more in importance. Major companies use artificial intelligence for their products [1]. This development is driven by advancements in methods such as deep learning [2], [3] that were originally inspired by concepts from neuroscience. Together with the availability of substantial computational performance, these methods enable complex machine learning applications, such as image [4] or speech recognition [5]. Specialized hardware can lower the cost of these methods in terms of energy, time, and therefore money [6], enabling either a scaling to larger problem sizes or the use in new devices outside of data centers.

Manuscript received February 17, 2016; revised May 13, 2016; accepted June 5, 2016. Date of publication September 9, 2016; date of current version January 26, 2017. This work was supported by the European Union Seventh Framework Programme ([FP7/2007-2013]) under Grant Agreement 604102 (HBP), 269921 (BrainScaleS), and 243914 (Brain-i-Nets). *(Simon Friedmann and Johannes Schemmel contributed equally to this work.)* This paper was recommended by Associate Editor G. Yuan.

The authors are with Kirchhoff Institute for Physics, 69120 Heidelberg, Germany (e-mail: schemmel@kip.uni-heidelberg.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBCAS.2016.2579164

On the other hand, using simulations of neural networks as a major tool for research in neuroscience depends on efficient simulators for large-scale networks. This opens the opportunity to build specialized hardware systems that serve as efficient platforms for research as well as technology. Multiple systems with this goal have been proposed, e.g., [7]–[10].

While the problem can be approached in different ways, the concept of analog neuromorphic hardware [11], [12] promises especially area and energy efficient solutions as demonstrated by, e.g., [13]–[15]. These systems use the concept of a physical model to emulate neural networks: the temporal development of the membrane voltages of the neurons is emulated by custom analog circuits, representing the neuron and synapses of the emulated network. However, neurons and synapses built this way are limited to at best a family of models that are compatible with their physical realization. On the other end of the spectrum, software allows the simulation of arbitrary models by solving numerical equations.

Especially, there exists a large set of different models for learning and plasticity, so that a flexible hardware implementation is desirable. This is true for technical applications where one network is often trained with different methods for pre-training and fine-tuning [3], as well as biology where different plasticity rules are found depending on cell type and brain region [16], [17]. But besides flexibility, efficiency is a key concern in both domains. Large-scale simulations have been demonstrated in the past [18]–[20], but, especially with plasticity, simulation time quickly becomes a limiting factor even on medium-sized networks [21]. Similarly, in the technical domain, significant effort is put into accelerating learning including the use of Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs) [6], [22].

For this study, we follow a novel hybrid approach to learning as a trade-off between efficiency and flexibility: we use full-custom analog circuits for real-time and parallel processing of spikes in the emulated synapses. These circuits serve as sensors for an embedded general-purpose processor that implements the learning rule in software. This way, we offer a solution that allows biologically realistic plasticity while emulating networks a thousand times faster than in biology. Using physical models for core components, this speed-up is not affected by network size or activity. In this study we present results from a scaled-down prototype that demonstrates for the first time plasticity in such a hybrid system using analog components together with an embedded Plasticity Processing Unit (PPU).

The study starts with a description of analog circuits and the architecture of the PPU in Section II. After that, we introduce

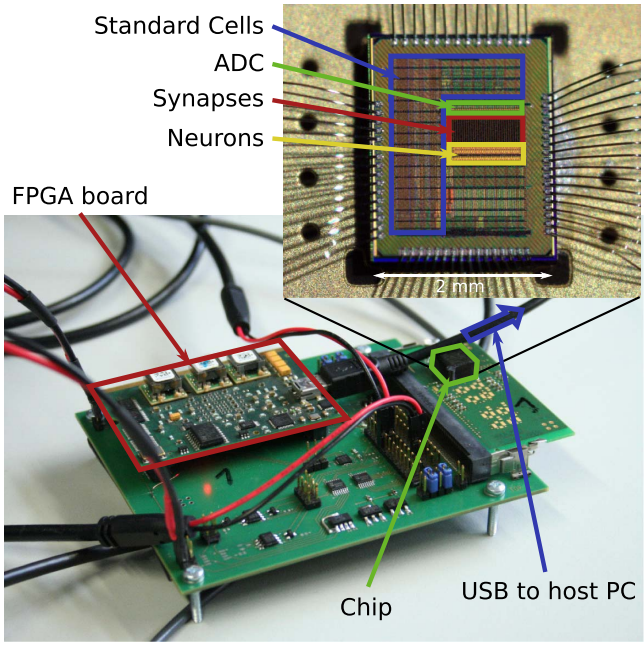

Fig. 1. Photograph of die and test system. The active die area is 1.7×2.2 mm^2. The host computer communicates via USB with an FPGA board. The FPGA controls Digital to Analog Converters (DACs) on the board for bias generation and communicates with the chip through a Serializer/Deserializer (SerDes) interface.

the theoretical background and methods in Section III. Then, results are presented for simulations in Section IV and for experiments in Section V. Finally, Section VI discusses results, followed by conclusion and outlook in Sections VII and VIII.

II. Description of Circuits

The circuits presented in this paper are part of a prototype ASIC for the next generation of a large Neuromorphic Hardware system [7]. All results have been measured using the setup shown in Fig. 1. The individual components of the chip and their functional relations are depicted in Fig. 2. The central elements are an array of 2048 synapses and 64 neuron-compartment circuits, which implement the analog, continuous-time emulation of their biological counterparts. Similar to the predecessor system described in [7] the presented chip operates faster than wall-clock time. To simplify the calibration of the analog elements to the model equations, the acceleration factor is fixed at 10^3. Therefore, one second in the model time scale is emulated in one millisecond by the presented system.

The focus of this paper is the plasticity sub-system, which observes the activity of the emulated neural network and modifies its parameters in reaction to these observations depending on the configured plasticity rule. The neuron circuits are not covered in this publication.

The plasticity sub-system is a mixed-signal, highly-parallel control loop simultaneously monitoring the temporal correlation between all pre- and post-synaptic firing times. The plasticity rule itself is implemented as software running on an embedded micro-processor, the PPU. It evaluates the signals from the analog correlation sensors located within the synapses and computes weight updates. Besides the synapse, it can

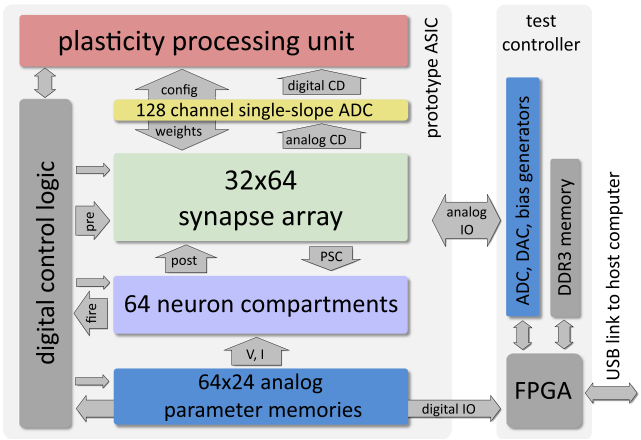

Fig. 2. Block diagram of the presented system. The prototype ASIC is shown to the left. A photograph of the system can be seen in Fig. 1. "CD" stands for "correlation data" and "PSC" for "post-synaptic current."

observe firing rates of neurons and modify parameters of the emulated neurons as well as the topology of the network. Connection to the outside world allows the integration of third factors, for example a reward signal [23].

The parallel analog implementation of the correlation sensors in every synapse allows the plasticity sub-system to handle the high rate of simultaneous events[1] The circuit maintains a local eligibility trace that depends on the relative timing of pre- and post-synaptic firing.

A 128 channel single-slope Analog to Digital Converter (ADC)[2] digitizes the stored trace information for the PPU.

A. Synapse

1) Basic Operation Principles: In Fig. 2 the synapses are arranged in a two-dimensional array between the PPU and the neuron compartment circuits. Pre-synaptic input enters the synapse array at the left edge. For each row, a set of signal buffers transmit the pre-synaptic pulses to all synapses in the row. The post-synaptic side of the synapses, i.e., the equivalent of the dendritic membrane of the target neuron, is formed by wires running vertically through each column of synapses.

At each intersection between pre- and post-synaptic wires, a synapse is located. To avoid that all neuron compartments share the same set of pre-synaptic inputs, each pre-synaptic input line transmits—in a time-multiplexed fashion—the pre-synaptic signals of up to 64 different pre-synaptic neurons. Each synapse stores a pre-synaptic address that determines the pre-synaptic neuron it responds to.

Fig. 3 shows a block diagram of the synapse circuit. The main functional blocks are the address comparator, the DAC and the correlation sensor. Each of these circuits has its associated memory block.

The address comparator receives a 6 bit address and a pre-synaptic enable signal from the periphery of the synapse array

[1]Due to the acceleration factor of 10^3 every component has to handle a thousandfold higher data rate as a comparable unaccelerated system operating at biological time scale.

[2]The initial design of the Analog to Digital Converter (ADC) was done by Sabanci University, Turkey.

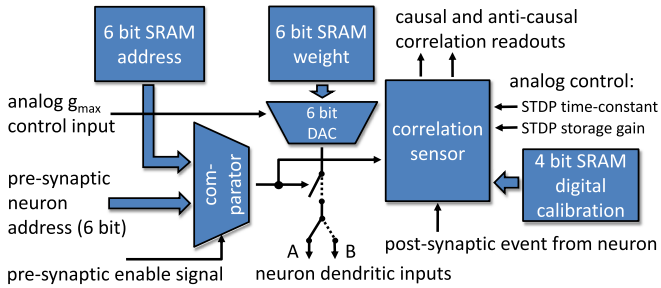

Fig. 3. Block diagram of the synapse circuit.

as well as a locally stored 6 bit neuron number. If the address matches the programmed neuron number, the comparator circuit generates a pre-synaptic enable signal local to the synapse (*pre*), which is subsequently used in the DAC and correlation sensor circuits.

Each time the DAC circuit receives a *pre* signal, it generates a current pulse. The height of this pulse is proportional to the stored weight, while the pulse width is typically 4 ns. This matches the maximum pre-synaptic input rate of the whole synapse row which is limited to 125 MHz. The remaining 4 ns are necessary to change the pre-synaptic address. The current pulse can be shortened below the 4 ns maximum pulse length to emulate short-term synaptic plasticity [24].

Each neuron compartment has two inputs, labeled A and B in Fig. 3. Usually, the neuron compartment uses A as excitatory and B as inhibitory input. Each row of synapses is statically switched to either input A or B, meaning that all pre-synaptic neurons connected to this row act either as excitatory or inhibitory inputs to their target neurons. Due to the address width of 4 bit the maximum number of different pre-synaptic neurons is 64.

The remaining block shown in Fig. 3 is the correlation sensor, which has a 4 bit static memory associated with it. Its task is the measurement of the time difference between pre- and post-synaptic spikes. To determine the time of the pre-synaptic spike it is connected to the *pre* signal. The post-synaptic spike-time is determined by a dedicated signaling line running from each neuron compartment vertically through the synapse array to connect to all synapses projecting to inputs A or B of the compartment. This signal, which is called *post* subsequently, has a similar pulse length as the *pre* signal.

The correlation sensor measures the causal (*pre* before *post*) and anti-causal (*post* before *pre*) time differences and stores them as exponentially weighted sums within the synapse circuit. In comparison to earlier implementations [14] by the authors the circuit has been improved in two main aspects: first, only one instance of the time measurement circuit is now re-used for causal as well as anti-causal time difference measurements, resulting in strongly reduced mismatch between the causal and anti-causal branches of the activation function. Second, the time-constant of the exponential is now truly adjustable over more than two orders of magnitude to fit most biological models of spike-time dependent plasticity [25].

Due to the implementation in a much smaller process feature size, 65 nm instead of 180 nm, four static memory bits could be allocated for additional calibration of transistor variations

TABLE I
KEY PARAMETERS OF THE SYNAPSE CIRCUIT

Parameter	Value
V_{dd} thin oxide	1.2V
V_{dd} thick oxide	2.5V
area	$94\mu m^2$
total # of MOSFET	205
$C_{causal,\ anti\text{-}causal}$	6fF, MOSCAP
$C_{transfer}$	3-9fF, MOSCAP, adjustable
$C_{storage}$	37fF, MIMCAP

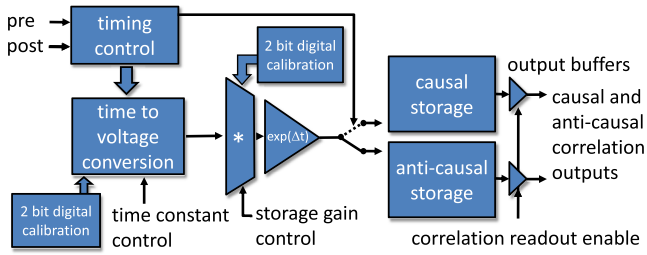

Fig. 4. Block diagram of the correlation sensor circuit.

within each synapse. Table I summarizes key parameters of the synapse implementation.

2) Correlation Sensor Circuit: The structure of the correlation sensor is shown in Fig. 4. The input stage receives *pre* and *post* signals and uses them to generate the internal timing. A time to voltage conversion circuit generates a voltage representing the elapsed time between the most recent *pre* and *post* events. This voltage is scaled by the storage gain parameter and the result is used as argument to an exponential function. This exponentially weighted time difference is added to one of two storage circuits. The selection of the storage circuit depends whether the last input event seen has been a *pre* or *post* signal. *Pre* before *post* is stored in the causal storage, *post* before *pre* in the anti-causal one.

To counteract the effects of fixed-pattern noise created by transistor variations, the time to voltage as well as the storage gain stages have two digital calibration inputs each. The four calibration bits are stored locally in each synapse. The time constant of the time to voltage conversion can be set for one row of synapses by a control voltage. The same applies to the storage gain stage, where the storage gain control signal adjusts one row of synapses. In the prototype chip the gain and time constant input signals of each row are shorted and connected to two external input pins.

The values stored in the causal and the anti-causal storage cells can be read out simultaneously for all synapses in a row. A parallel single-slope ADC at the top of the synapse array converts the analog values read out from the storage cells into digital words for the PPU (see Fig. 2).

Fig. 5 depicts the correlation sensor circuit. To enhance the readability of the circuit diagram, the individual blocks of Fig. 4 are not marked. See the caption for assignments of the components to the different functional blocks.

As stated above, the correlation sensor monitors the temporal correlation between *pre* and *post* synaptic firing events. This is accomplished by charging the capacitors C_{causal} and $C_{\mathrm{anti-causal}}$ with a constant current. The selection of the capacitor depends on the temporal order of the pre and post signals.

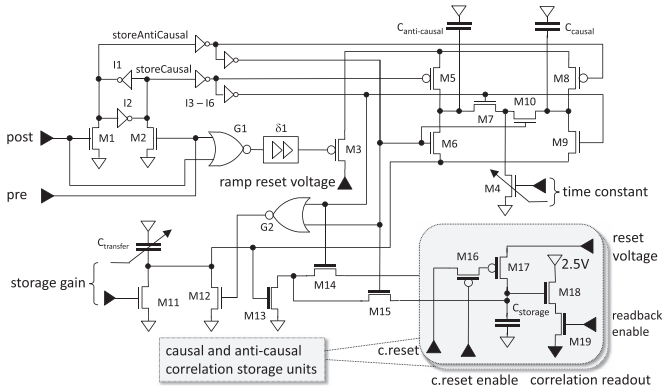

Fig. 5. Simplified circuit diagram of the correlation sensor. All supply connections are 1.2 V if not stated otherwise. A thick gate symbol depicts a thick gate-oxide transistor capable of 2.5 V operation. The assignment of the components to the functional blocks depicted in Fig. 4 is as follows: timing control—M1-2, I1-6, G1-2, $\delta 1$; time to voltage conversion—M3-10, C_{causal}, $C_{\mathrm{anti-causal}}$; exponential—M13; storage gain—M11-12, C_{transfer}; storage—M14-17; storage output buffer—M18-19, C_{storage}.

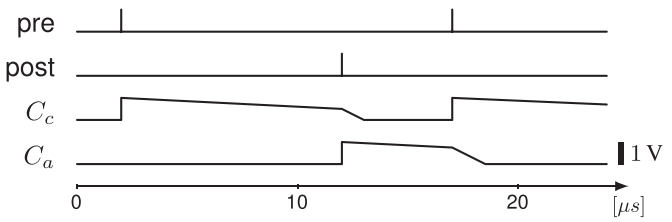

Fig. 6. Exemplary timing of the correlation sensor circuit. The timescale is of the order of the correlation sensor time constant τ_c.

As can be seen in Fig. 6, the arrival of a *pre* pulse starts the charging of C_{causal} after discharging it quickly to its initial value, while $C_{\mathrm{anti-causal}}$ starts charging after a *post* pulse. Two or more *pre* or *post* pulses in succession would only restart the discharge/charge process without changing the capacitor. Therefore, the correlation sensor only supports plasticity rules based on nearest neighbor schemes [26].

To determine the temporal order, the input stage of the correlation sensor utilizes a D-latch formed by I1 and I2. Each time a *post* follows a *pre* or vice-versa, the D-latch gets toggled by M1 or M2, respectively. To orchestrate the precise discharging and switching of capacitors within the limited area of the synapse, the circuit makes use of the delays of the individual components. In Fig. 8 a subset of the relevant signals is shown. The inverters I1 and I2, which form the D-latch, have a very low drive strength. This leads to a significant delay between the internal node being discharged by an external *pre* or *post* pulse, and the respective inverted internal node (*storeAntiCausal* in case of a *pre* pulse or *storeCausal* after a *post* signal).

This time difference is used by G2 to produce a short pulse at the gate of M12 to precharge C_{transfer} (see below). The current charging the capacitors C_{causal} and $C_{\mathrm{anti-causal}}$, and therefore controlling the time constant of the correlation sensor, is generated by an adjustable current sink M4. The gate voltage of M4 is shared by all synapses of a row. To reduce the fixed pattern noise within a synapse row, the length of M4 can be digitally controlled in four steps by approximately 20%. This allows to

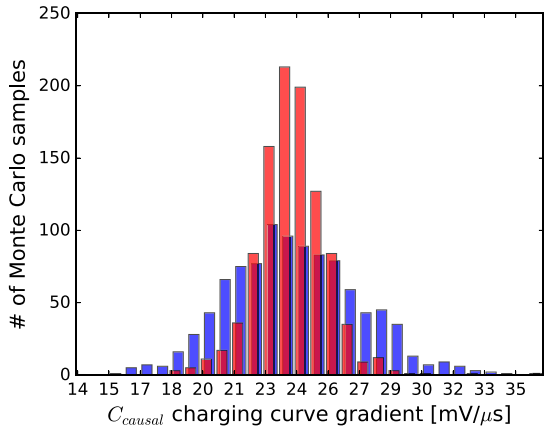

Fig. 7. Results of a Monte-Carlo simulation showing the effect of the two-bit digital time-constant calibration built into each synapse. The two histograms show the distribution of the gradient of the charging curve of C_{causal} (dashed trace in Fig. 9) over 1000 MC-runs each. The uncalibrated case is shown in blue, the calibrated in red.

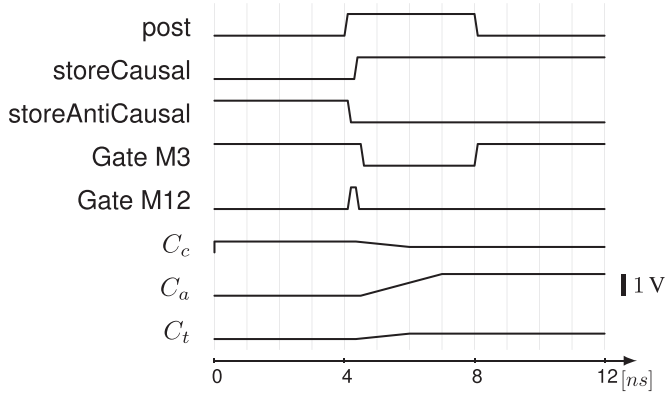

Fig. 8. Timing of the correlation sensor circuit: zoom-in on the time axis around the post event shown in Fig. 6. The following abbreviations are used for the capacitor labels: $C_{\mathrm{c(ausal)}}$, $C_{\mathrm{a(nti-causal)}}$, and $C_{\mathrm{t(ransfer)}}$.

reduce the fixed pattern noise by selecting for each synapse the value which minimizes synapse to synapse variation within the row. Fig. 7 shows the results of a Monte-Carlo simulation demonstrating the effectiveness of this approach.

In the full-size neural network chip each row will have an individual bias generation for M4, which allows different time constants in different rows, as well as the calibration of the row mean of the time-constant. The presented prototype chip directly connects all time-constant inputs to an external input pin which is driven by the test controller (see Fig. 2).

The state of the D-latch determines whether C_{causal} or $C_{\mathrm{anti-causal}}$ is charged by M4 through the inverter chains formed by I3 to I6 and M7 as well as M10.

The subsequent discussion is based on the temporal relations depicted in Fig. 8. As can be seen in Fig. 6, the charging process of C_{causal} or $C_{\mathrm{anti-causal}}$ starts after it has been discharged to the ramp reset voltage by the *pre* or *post* event. In the case shown in Fig. 8, $C_{\mathrm{anti-causal}}$ is discharged. The initial discharge is initated in two steps: first, after arrival of a *post* pulse, $C_{\mathrm{anti-causal}}$ is connected to M3 by enabling M5. The enabling of M3 is delayed to make sure the other capacitor, C_{causal} is

disconnected from M3 by M8. This is essential since at this moment C_{causal} holds the last causal time-difference measurement result which should not be altered by the discharge of $C_{\text{anti-causal}}$. At the beginning of the *post* pulse the voltage on C_{causal} is as follows:

$$V_{C_{\text{causal}}}^{\text{begin post}} = V_{\text{ramp reset}} - \frac{I_{\text{M4}} \cdot (t_{\text{post}} - t_{\text{pre}})}{C_{\text{causal}}}. \quad (1)$$

After a *pre* pulse a similar equation holds for the voltage on $C_{\text{anti-causal}}$

$$V_{C_{\text{anti-causal}}}^{\text{begin pre}} = V_{\text{ramp reset}} - \frac{I_{\text{M4}} \cdot (t_{\text{pre}} - t_{\text{post}})}{C_{\text{anti-causal}}}. \quad (2)$$

The intial discharge process finishes within the time-interval set by the length of the *post* pulse. After *post* becomes inactive, M3 is deactivated and the charging of $C_{\text{anti-causal}}$ by the current flowing through M7 and M4 starts. Simultaneously, the transfer of the causal result from C_{causal} to the storage capacitor $C_{\text{storage causal}}$ is initiated. In Fig. 5 only one of the two identical storage circuits is drawn. Depending on the state of the *storeCausal* and *storeAntiCausal* signals, M14 or M15 connect one of the storage circuits to M13. The timing of these signals assures that M14 and M15 are never activated simultaneously.

The charge transfer starts by enabling M9, thereby connecting C_{causal} to C_{transfer}. To avoid any crosstalk from the previous transfer process, M12 is always activated prior to M9 and charges C_{transfer} to V_{dd}. After M9 is enabled, charge charing between C_{causal} and C_{transfer} starts. The charging process will be completed before *post* becomes inactive, but C_{causal} and C_{transfer} will stay connected until the end of the storage cycle.

After the *post* pulse, before the charging of C_{transfer} and C_{causal} starts, the voltage on C_{transfer} can be calculated as follows:

$$V_{C_{\text{transfer}}}^{\text{end post}} = \frac{V_{C_{\text{causal}}}^{\text{begin post}} C_{\text{causal}} + V_{C_{\text{transfer}}}^{\text{begin post}} C_{\text{transfer}}}{C_{\text{transfer}} + C_{\text{casual}}}. \quad (3)$$

Since C_{transfer} has been charged by M12 at the very beginning of the *post* pulse, $V_{C_{\text{transfer}}}^{\text{begin post}}$ is zero and (3) simplifies to

$$V_{C_{\text{transfer}}}^{\text{end post}} = \frac{V_{C_{\text{causal}}}^{\text{begin post}} C_{\text{causal}}}{C_{\text{transfer}} + C_{\text{casual}}}. \quad (4)$$

The capacitance of C_{transfer} is adjustable in four steps to allow the reduction of synapse-to-synapse variations.

Fig. 9 shows a simulation of the charging process of C_{causal} and C_{transfer} by M11. After the *post* pulse, as long as the *storeCausal* signal is active, C_{causal} and C_{transfer} are connected by M9 and their voltages are equal. The charging current is set by the gate voltage of M11. In the presented prototype chip, this voltage is directly connected to an analog input pin and set by the test controller (see Fig. 2).

Before the time difference is stored for a causal or anti-causal measurement, its exponential value has to be calculated. This is accomplished by M13. While M13 is connected to one of the storage capacitors C_{storage} by M14 or M15, it discharges the respective storage capacitor. The amount of charge it can remove from C_{storage} depends on its gate voltage, which follows the time course shown in Fig. 9.

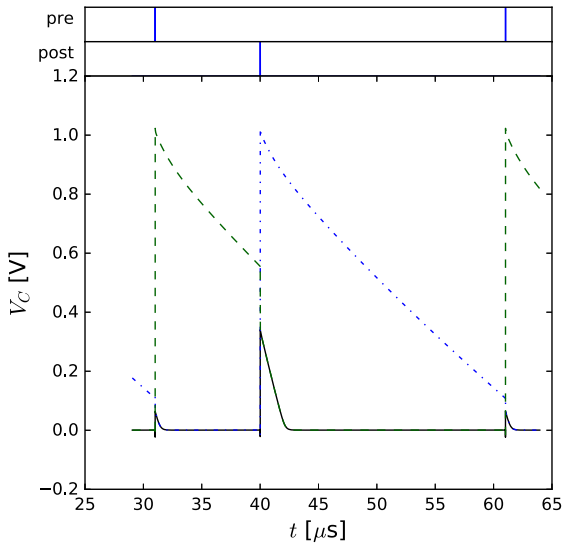

Fig. 9. Simulation showing the charging of C_{transfer} (solid trace) after a *post* pulse (at $t = 40$ μs). The dashed trace shows the voltage on C_{causal} and the dashed-dotted trace the voltage on $C_{\text{anti-causal}}$. During the charging process, the appropriate C_{storage} capacitor is discharged.

The purpose of the charge sharing between C_{causal} and C_{transfer} is the reduction of the voltage representing the measured time difference below the threshold voltage of M13. This ensures the operation of M13 in weak inversion. Therefore, we can use the sub-threshold model to calculate the current through M13 at any time t

$$I_{DS}(t) = \frac{W}{L} I_{D0} \exp\left(\frac{V_{GS}(t)}{nkT/q}\right) \quad (5)$$

$$V_{GS}(t) = V_{C_{\text{transfer}}}(t) \quad (6)$$

$$I_{DS}(t) = I_{C_{\text{storage}}}(t). \quad (7)$$

Since $V_{C_{\text{transfer}}}(t)$ changes after C_{transfer} has been discharged to its initial voltage during the *post* pulse, $V_{C_{\text{transfer}}}^{\text{end post}}$, (5) has to be integrated over the time interval from the *post* pulse, t_p, to t_e, the point in time when $V_{C_{\text{transfer}}}(t)$ has been charged completely, i.e., $V_{C_{\text{transfer}}}(t)$ is close to zero

$$\Delta Q_{C_{\text{storage}}} = \int_{t_p}^{t_e} I_{C_{\text{storage}}}(t) dt. \quad (8)$$

To solve this integral a simple linear model is used for the charging of C_{transfer} from $V_{C_{\text{transfer}}}^{\text{end post}}$ to zero

$$V_{C_{\text{transfer}}}^{\text{end post}} = V_{C_{\text{transfer}}}(t), t = t_p \quad (9)$$

$$V_{C_{\text{transfer}}}(t) = V_{C_{\text{transfer}}}^{\text{end post}} \cdot \left(1 - \frac{t - t_p}{t_e - t_p}\right), t_p \leq t \leq t_e. \quad (10)$$

The time difference $t_e - t_p$ can be calculated from the current through M11 and the involved capacitances as follows:

$$t_e - t_p = \frac{V_{C_{\text{transfer}}}^{\text{end post}} \cdot (C_{\text{transfer}} + C_{\text{casual}})}{I_{\text{M11}}}. \quad (11)$$

Solving (8) gives

$$\Delta Q_{C_{\text{storage}}} = \frac{W}{L}\frac{nkT}{q} I_{D0} \frac{t_e - t_p}{V_{C_{\text{transfer}}}(t_p)} \exp\left(\frac{V_{C_{\text{transfer}}}(t_p)}{nkT/q} - 1\right). \tag{12}$$

Using the result of (12) the change in the voltage stored on C_{storage} can be calculated

$$\Delta V_{C_{\text{storage}}} = \frac{\Delta Q_{C_{\text{storage}}}}{C_{\text{storage}}}. \tag{13}$$

For typical values of the transfer gain, which controls $t_e - t_p$ by setting I_{DS} of M11, the deviation between (12) and the ideal exponential activation function is below 1%. Also, due to the exponential decay of $I_{C_{\text{storage}}}(t)$, only the very first part of the charging of C_{transfer} contributes to $\Delta V_{C_{\text{storage}}}$ significantly. If the discharge of C_{transfer} is interrupted by an arriving *pre* pulse, the resulting error is minimal.

No control signal is needed to end the charging of C_{transfer}, avoiding any distortions caused by clock-feedthrough. The current $I_{C_{\text{storage}}}(t)$ is reduced to the minimum sub-threshold current without negative gate overdrive as $V_{GS_{\text{M13}}}$ approaches 0 V. Since M13 is a thick oxide transistor with a long gate, this current is below 1 nA. Measured total leakage on C_{storage} was 1.7 mV/ms at 50° and only 0.14 mV/ms at room temperature (approx. 25°). The usable dynamic range of $V_{C_{\text{storage}}}$ is 1.3 V.

M13 together with M14 or M15, respectively, also protect the thin oxide transistors used in the time difference measurement circuits from the higher supply voltage of the storage circuits. To reach sufficient storage times the utilization of thick oxide transistors is necessary to avoid gate tunneling currents. The gate voltage of M14 and M15 comes from the thin oxide supply voltage, thereby limiting their source voltages to save values.

As a second function M14 and M15 act as cascodes to limit the voltage swing at the drain of M13, thereby reducing the variation of $\Delta V_{C_{\text{storage}}}$ as a function of the stored voltage on C_{storage}.

The storage circuits themselves use MIM-capacitors as storage cells, sitting on top of the each synapse, whereas C_{causal}, $C_{\text{anti-causal}}$, and C_{transfer} are implemented as MOS-capacitors. C_{transfer} uses several individual transistors to accomplish the digital calibration feature.

Each storage circuit uses a source follower (M18) for the readout of the stored correlation results. A pass-transistor (M19) connects the source follower to the correlation readout line if the readback enable signal of the row is active. There are two readout lines per synaptic column, thereby causal and anti-causal data of every synapse in one row can be simultaneously connected to the inputs of the correlation ADC at the top of the synapse array.

Each storage capacitor of the synapse array can be cleared individually by activating a causal or anti-causal column correlation reset signal together with a row-wise correlation reset enable. During network operation the PPU generates a pattern on the correlation reset inputs, depending on the results of the plasticity calculations, before it applies the column reset enable. The reset voltage can be adjusted, as can the bias current of the readback source followers, to adjust the readback voltage range to the input range of the correlation ADCs.

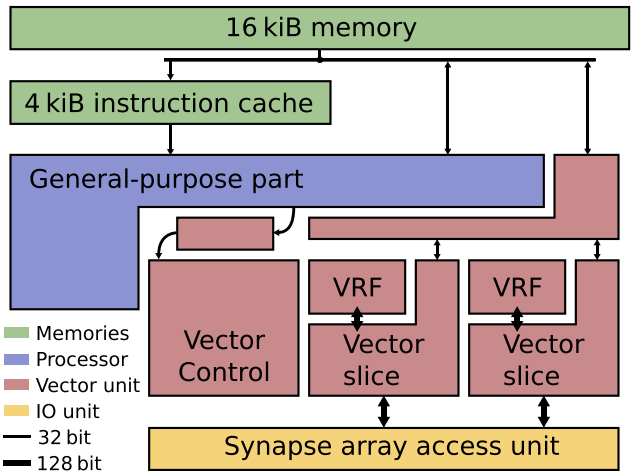

Fig. 10. The PPU is part of the plasticity sub-system and computes weight updates. It consists of a general-purpose part implementing the Power ISA and a special-function unit to accelerate computations using Single Instruction Multiple Data (SIMD) operations. The processor has access to 16 kiB of on-chip memory and uses a 4 kiB direct-mapped instruction cache. The special-function unit consists of a shared control unit for multiple datapath slices operating on 128 bit vectors. See Fig. 11 for details of the vector unit.

B. Plasticity Processing Unit

Fig. 10 shows an overview of the PPU. It is a general-purpose micro-processor extended with a functional unit specialized for parallel processing of synapses. The general-purpose part implements the Power ISA 2.06 [27] in order to be compatible with existing compilers. We have chosen a 32 bit embedded implementation. Instructions are issued in order and can retire out of order. The core does not have a floating-point unit, but includes fixed-point hardware multiplier and divider. In the presented chip it has access to 16 kiB of main memory with a direct-mapped instruction cache of 4 kiB. The SystemVerilog source code of the implementation is available as open source from [28].

The special-purpose functional unit implements an instruction set extension for accelerated processing of synapses. Following the SIMD principle a single control unit operates multiple—two for the presented chip—datapath slices. Each slice operates on 128 bit wide vectors of either eight or sixteen elements. Of these vectors 32 can be stored in a dedicated register file in each slice. Fig. 11 shows a block diagram of the unit.

The vector unit is organized as a weakly coupled co-processor with five functional units that have their own reservation stations. Upon encountering vector instructions, the general-purpose part sends them to a queue, which completes execution on the general-purpose side. The vector unit takes instructions in order from this queue, decodes them and distributes them to the appropriate functional units.

The five functional units provide operations for arithmetics, comparison, permutation, load/store from main memory, and load/store from synapses. Table II lists what types of operations are implemented. All operations are available in two modes treating their operands either as vectors of sixteen 8 bit or eight 16 bit elements. This allows trade-offs between throughput and accuracy and is also necessary to support the capability of combining synapses to achieve weights of higher resolution.

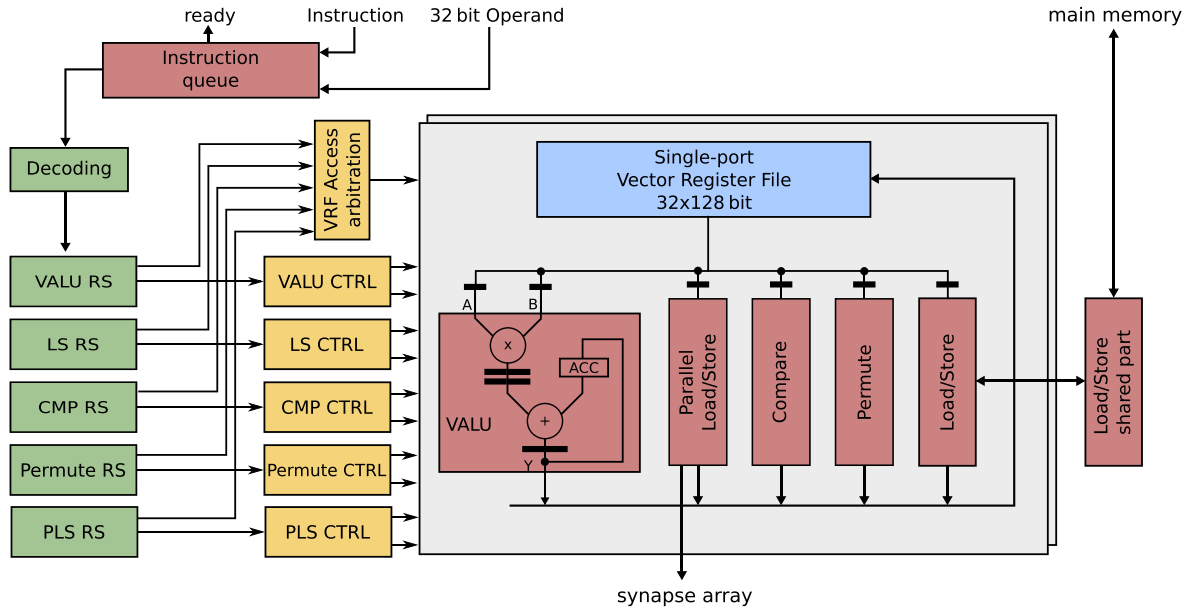

Fig. 11. Detailed view of the special-function unit for SIMD operations within the PPU. The general-purpose part sends instructions with optionally a 32 bit operand from the general-purpose register file via a queue. The decoding unit dispatches instructions to the respective reservation station upon availability. When operands are available and the execution unit is ready, the reservation station issues the operation to the control unit, which controls the multiple parallel datapaths. The vector register file has a single port for reading and writing. Access is arbitrated between reservation stations using a pseudo-random fair scheme. The serial load/store unit accesses main memory through a shared datapath.

TABLE II
IMPLEMENTED OPERATIONS

Category	Operations
Modular 16 bit	mult-acc, mult, add, sub, cmp
Modular 8 bit	mult-acc, mult, add, sub, cmp
Saturating 16 bit fractional	mult-acc, mult, add, sub
Saturating 8 bit fractional	mult-acc, mult, add, sub
Permutation	select, shift, pack, unpack
Load/store	load/store parallel, load/store serial

A minimum of 8 bit is required, since the ADC uses that particular resolution. In addition to the two modes of different size, vector elements can be treated either to be in signed integer or signed fractional representation. For the latter case saturating arithmetic is used, while integers always use modular arithmetic. The arithmetic functional unit is centered around a fused multiply-add data path, which also executes instructions for simple addition, subtraction, and multiplication.

The comparison unit writes results to a vector condition register holding flags for equality, less than, and greater than for each byte. These flags can be used by a select operation provided by the permutation unit to selectively combine two registers into one depending on a previous compare operation. Also, arithmetic and load/store operations support conditional execution using the vector condition register. Further operations provided by the permutation unit are bit-shifting, loading vectors from general-purpose registers, and conversion between fractional 16 bit and storage representation.

The two load/store units serve different purposes: one is meant for initialization of vector registers by sequentially loading words of 32 bit length from main memory. The other uses a fully parallel bus for accesses on synapses and the ADC. In the presented chip this bus has a width of 256 bit.

C. Input/Output With Analog Part

A specialized Input/Output (IO) unit translates the load and store operation on the parallel bus into transactions to the appropriate blocks based on the used address. Potential targets are synapse memory, ADC, and correlation readout. Typically, the PPU will iterate over all rows of synapses sequentially reading weights and correlation data and writing back updated weights. Therefore, the access unit allows multiple transactions to be in progress simultaneously. For example, performing a Static Random Access Memory (SRAM) read operation, while an analog-to-digital conversion of correlation data is ongoing.

The presented chip can process 32 synapses in parallel, when using byte-mode operations. Therefore, it takes two steps to compute updates for a full row of 64 synapses. Since IO operations work on full rows, the access unit supports buffering: results are kept in the output registers of analog blocks after a read transaction completes. If the next read refers to the same row, the buffered results are returned immediately.

The access unit also executes requests from outside of the chip performed through a 32 bit wide bus. Arbitration with PPU accesses uses a pseudo-random fair scheme: a flip-flop indicates which requester is favored upon conflict. For every conflict the state of the flip-flop is inverted.

D. Considerations for Plasticity Processing

The architecture includes several design decisions geared towards the main use-case of computing weight updates. Synaptic plasticity models from biology are typically local to the synapse, i.e., synapses can be computed independently. This is true for classical Spike-timing dependent plasticity (STDP) models [17], [26] and many phenomenological models

[29]–[33]. Therefore, parallel processing of synapses is viable and we realize this using the SIMD approach.

The vector unit is weakly coupled to the general-purpose part of the processor: the two parts do not synchronize instruction execution or share instruction tracking logic. Only when the instruction queue is full, does the general-purpose part stall. This allows to overlap execution in both parts to a large extent. The general-purpose part is primarily concerned with control-flow and sends the plasticity kernel to the vector unit as a stream of instructions.

For the execution of the plasticity kernel it is important, that IO accesses and computation are pipelined to achieve good performance. While new weights for the current row of synapses are computed, the ADC should simultaneously convert analog values for the next row. To achieve this in an efficient and automatic way, we use reservation stations for out of order execution of vector operations. Each functional unit shown in Fig. 11 has a reservation station (shown in green). Within one reservation station instructions are issued in order.

Implementing several reservation stations is more costly than following a simpler scheme for in-order issue as it is done in the general-purpose part. Because control logic is shared for all vector slices, this additional cost does not impinge on scalability to larger synapse arrays. On the other hand, area of the vector slices themselves has to be minimized. This reflects for example in the use of a single-port register file instead of a more typical three-port variant. Thereby, register access is a bottleneck for execution—an instruction will typically read two operands and write one result requiring three cycles on the register file—that has to be minimized. Therefore, we opted for a multiply-accumulate unit with internal accumulator, so that multiplication and summation can be done in one instruction and instructions can be chained without dependency on the register file.

Apart from that, we selected a minimal set of instructions focusing on fixed-point arithmetics and IO operations to save area in the vector slices. The only concession are pack and unpack operations as part of the permute unit to efficiently convert between weight representations for storage and computation (see Section III-B).

To save power while plasticity is not needed at all or waiting for the next update cycle, the clock of the PPU is gated. The clock is disabled when the PPU enters the sleep state by executing the Power ISA wait instruction. Any interrupt request, for example from a timer or an external request, re-enables the clock and wakes the processor up.

III. Theory and Methods

Fig. 12 shows the experimental protocol used for simulations with the PPU and all later measurements in hardware. The synapse is stimulated with presynaptic spikes at times $X_i = 0, T, 2T, \ldots, NT$ where T is the interspike interval and N is the total number of spikes. Postsynaptic spikes are shifted by Δt giving firing times $Y_i = \Delta t, T + \Delta t, 2T + \Delta t, \ldots, NT + \Delta t$. The synapse circuit accumulates this stimulation into the two local traces a_+ and a_- as described in Section II-A. The ADC converts the analog traces into 8 bit digital values A_+

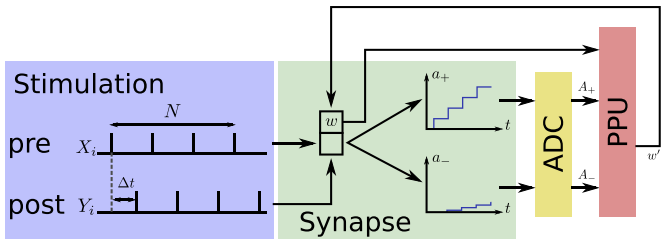

Fig. 12. Protocol for single synapse experiments. Regular spike trains with a relative shift of Δt are sent to the pre and post inputs of the correlation measurement circuit in the synapse. The local traces $a_\pm$ are read out using the ADC. For experiments reported in Section V-D the PPU computes new weights.

and A_-, respectively. These values together with the synaptic weight w are the input for the PPU that computes the new weight w'.

For this study we use a multiplicative STDP rule as reference model (see for example [26])

$$w' = w + \begin{cases} \lambda(w_{\max} - w)\exp\left(-\frac{\delta}{\tau_+}\right) & \text{for } \delta > 0 \\ -\lambda\alpha w \exp\left(\frac{\delta}{\tau_-}\right) & \text{for } \delta \leq 0. \end{cases} \tag{14}$$

Here, λ is a scaling parameter, $w_{\max}$ is the maximum weight, δ is the time difference between pre- and postsynaptic firing ($\delta > 0$ if the presynaptic event occurs before the postsynaptic one), $\tau_\pm$ are time constants, and α controls the asymmetry between the pre-before-post ($\delta > 0$) and post-before-pre ($\delta < 0$) branches.

The exponential term in (14) is realized by the synapse circuit itself (see Section II-A) and accumulated on the local traces $a_\pm$. The $a_\pm$ correspond to the voltage on C_{storage} in the synapse circuit. We use a different symbol here to refer to the value visible to the PPU, i.e., including offset from the source follower of the readout circuit. The $a_\pm$ are also inverted compared to the physical voltage, so that $a_\pm = 0$ V corresponds to the reset value on C_{storage}. In an idealized model of the actual circuit, these traces are given by summing over previously observed spike-pairs

$$a_+ = \sum_{\text{pre}-\text{postpairs}} \eta_+ \exp\left(-\frac{\delta}{\tau_+}\right) \tag{15}$$

$$a_- = \sum_{\text{post}-\text{prepairs}} \eta_- \exp\left(\frac{\delta}{\tau_-}\right) \tag{16}$$

with the analog accumulation rates $\eta_\pm$. The summed up pairs are selected according to a reduced symmetric nearest neighbor pairing rule as defined in [26]. This is the same pairing scheme as was already used in [14]. To approximate the rule described by (14), the PPU uses the converted digital values $A_\pm$ to compute

$$A = A_+ - A_- \tag{17}$$

$$w' = w + \begin{cases} \lambda(w_{\max} - w)A & \text{for } A > 0 \\ \lambda\alpha w A & \text{else.} \end{cases} \tag{18}$$

After the update, the accumulation traces $a_\pm$ are reset to zero.

A. STDP Interaction Box

To quantify the measured STDP curves we extract two measures from the observed $a_\pm(\Delta t)$ dependency: the amplitude $\hat{a}_\pm$ and the full width at half maximum (FWHM) $\hat{\tau}_\pm$. For illustration they are plotted together as a box with height $\hat{a}_\pm$ and width $\hat{\tau}_\pm$ in Fig. 15. The amplitude is given as

$$\hat{a}_\pm = \max a_\pm - \min a_\pm. \tag{19}$$

FWHM is given as the range where $a_\pm$ is below $(1/2)(\max\ a_\pm - \min\ a_\pm) + \min\ a_\pm$.

B. Bit-Representation of Weights

Each synapse provides 6 bit of SRAM memory for weight storage. Two synapses can be combined to increase the effective weight to 12 bit. The PPU uses either 8 bit or 16 bit operations giving some freedom in how weights are represented for computation. For this study, we use a fractional number format with saturating arithmetic, i.e., over- and underflows are prevented by saturating to maximum and minimum values [34]. Weights are aligned to use the range from 0 to 1, i.e., one zero bit is added to the right for 8 bit computations as follows:

7							0
-1	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
0	w_5	w_4	w_3	w_2	w_1	w_0	0

Here, the w_i are the individual bits of the weight with w_5 being the most significant bit (MSB). For 12 bit weights the representation is as follows:

15							8
-1	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
0	w_{11}	w_{10}	w_9	w_8	w_7	w_6	w_5

7							0
2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
w_4	w_3	w_2	w_1	w_0	0	0	0

Bits $w_{11} \ldots w_7$ are physically stored in one synapse, while $w_6 \ldots w_0$ reside within the other one. Special pack and unpack operations are implemented to facilitate conversion between the shown representation for computation and the stored representation.

Since for this study synaptic transmission of events to the neuron is not used, weights are permanently kept in a vector register. So no IO operations are performed.

IV. Simulations

To quantify the inaccuracies added by weight resolution and numerical precision of computations performed by the PPU, we simulate the protocol outlined in Fig. 12 and Section III with an idealized synapse circuit and ADC. This means, that accumulation by the synapse follows equations (15) and (16) exactly. The PPU computes weight updates according to (17) and (18) using 8 bit mode for 6 bit weight resolution and 16 bit mode for 6 bit and 12 bit weight resolutions.

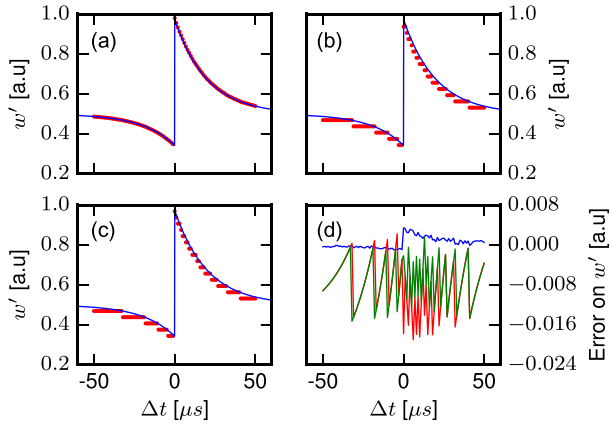

Fig. 13. Simulation results for weight updates with idealized synapses and ADC. The red points show the weight w' as computed by the PPU after stimulation. The blue lines show the result w'_{theo} with perfect precision. (a) 16 bit computational mode for 12 bit weight resolution. (b) 8 bit computational mode for 6 bit weight resolution. (c) 16 bit computational mode for 6 bit weight resolution. (d) Error $w' - w'_{\text{theo}}$ introduced by limited numerical precision. (blue: A, red: B, green: C).

A. Numerical Accuracy

Fig. 13 shows results for $N = 32$ spike-pairs with time-constants $\tau_\pm = 20\ \mu$s and accumulation rates $\eta_\pm = 0.25$ V. Weights are computed with $\lambda = 0.4$, $w_{\max} = 1$, and $\alpha = 1$. The initial weight is $w = 0.5$. The blue curves show the predicted result for updates performed without limited numerical precision based on the accumulated values $a_\pm$. The residuals shown in Fig. 13(d) therefore represent the error introduced by discretization of $a_\pm$ to 8 bit values $A_\pm$ in the ADC and numerical errors introduced by fixed-point arithmetic. This error is generally small: below 3.4×10^{-3} for Fig. 13(a), below 1.9×10^{-2} for Fig. 13(b), and below 1.5×10^{-2} for Fig. 13(c).

Notably, 6 bit weights systematically are smaller than predicted. According to these results, the use of the 16 bit mode for 6 bit synapses reduced the error especially for large updates, i.e., small $|\Delta t|$.

B. Updating Performance

The simulation used for the previous section also provides performance results in terms of achievable update rates. Depending on the learning task a minimum update rate may be required for correct functionality [35]. The classical model of STDP assumes immediate updates to the weight and so any delay can lead to mismatch to software simulations. Table III shows performance results for four different scenarios with and without ADC conversions and for different weight resolutions. The number of cycles represents the total time to update the full array of synapses. Row time is the resulting duration for a single row assuming a clock frequency of 500 MHz. The biological update rate shows the frequency of updates as seen by a single synapse translated into the biological time domain. The latter number assumes, that the update program iterates over all rows updating synapses in turn and is therefore a worst-case estimate.

The update frequencies are in all cases high compared to spike frequencies in the range of approximately 1–15 Hz expected from biology [36]–[38]. A previous study has identified

TABLE III
UPDATE RATES IN SIMULATION

No.	ADC	Mode	Resolution	Cycles	Row time	Bio. rate
1.	no	8 bit	6 bit	5122	320 ns	97.6 Hz
2.	no	16 bit	12 bit	4074	255 ns	122.7 Hz
3.	yes	8 bit	6 bit	11957	747 ns	41.8 Hz
4.	yes	16 bit	12 bit	6245	390 ns	80.1 Hz

1 Hz as a lower threshold for a particular correlation detection task [35]. However, their updating mechanism did not use an ADC but only employed a threshold comparison leading to larger errors on the accumulation traces $a_\pm$ for longer delays. It is, therefore, conceivable that for the same task the PPU-based approach is less sensitive to update frequency.

The ADC requires 560 ns for the conversion of one row of synapses. Rows 1 and 2 in Table III show that all other operations can execute in less time. Therefore, conversion by the ADC limits the update rate. Updates for 12 bit weights are generally faster, because two rows of synapse circuits are combined into one logical one. This leads to half the number of ADC conversions and computational operations. The additionally required pack and unpack operations to convert between stored and logical representation (see Section III-B) do not impact performance.

V. EXPERIMENTS

Fig. 1 shows the produced chip and the test setup used for experiments. The chip contains 64 neurons with 32 synapses each for a total of 2048 synapses. A single-ended SerDes link provides communication with a Xilinx Spartan-6 FPGA for control and event data. Link and internal logic operate with the same clock signal provided via a chip pin. The system is designed for frequencies up to 500 MHz and operated at 97.5 MHz in this study.

The FPGA is equipped with 512 MiB of DDR3-SDRAM and communicates with a PC via USB 2.0. Due to the real-time nature of neuromorphic hardware and the small time-scales involved, communication with the chip is buffered in the on-board SDRAM attached to the FPGA and played-back under precise timing control. The FPGA uses a byte-code with instructions of variable length to provide efficient coding with 64 bit effective time stamp resolution. The byte-code is executed at a clock frequency of 97.5 MHz leading to a best-case temporal precision of 10.26 ns. Responses and events are recorded with annotated timing information using the same byte-code representation.

A. Weight Linearity

We first analyze the DAC within the synapse. Fig. 14 shows the average output current for a total of 96 synapses on one chip over the full range of 64 possible weight values. The current was measured by sending a high-frequency input spike train to the synapse and measuring the resulting current using a readout pin and an external current measurement device.[3]

The fit yields an offset of 22.79 nA and a value of 11.52 nA for one least significant bit (LSB). With these values

[3]Keithley SourceMeter 2635

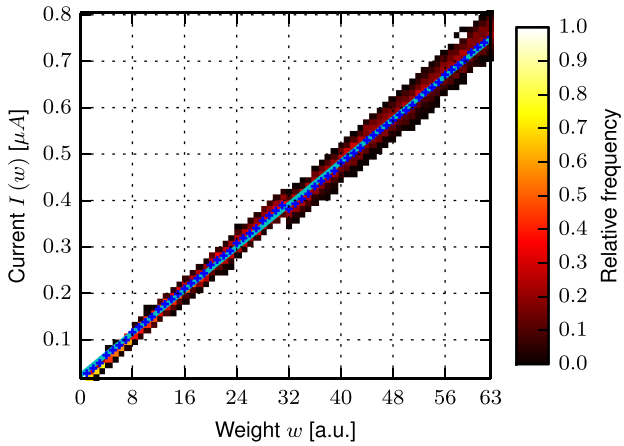

Fig. 14. Output current from the DAC within the synapse. The measurement includes 96 synapses (three columns) from one chip. Blue crosses mark the mean values. The best fit to all data points is shown as cyan colored line.

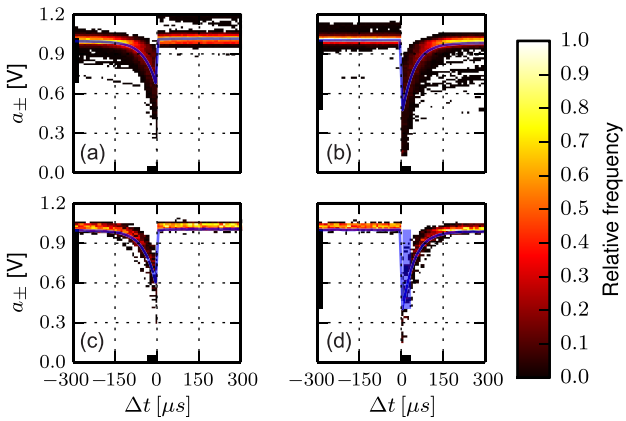

Fig. 15. Accumulation values $a_\pm$ after stimulation with $N = 32$ spike-pairs as two-dimensional histogram. Color indicates the relative frequency. The mean values are plotted as blue lines. The black bars on the axes indicate width and amplitude of the STDP interaction box (see Section III-A), which is also shown in blue in the last picture. Data points are shifted to an offset without stimulation of 1.00 V to correct for different offsets on different chips. (a) Post-before-pre measurement a_- for 672 synapses on three different chips. (b) Pre-before-post measurement a_+ for 800 synapses on three different chips. (c) and (d) Data from 32 synapses on the same ADC channel on one chip.

the maximal integral nonlinearity (INL) is 4.83 LSB, while the mean INL is 1.06 LSB. The systematic shift at the transition from code 31 to 32 is caused by well-proximity effects. Two fingers of the MSB transistor of the DAC are too closed to an adjacent well. This was only discovered after tape-out.

B. Variability

Fig. 15 shows the measured dependency $a_\pm(\Delta t)$ using the experimental protocol illustrated in Fig. 12. The curves were measured using $N = 32$ spike pairs and analog parameters $V_{\text{ramp}} = 250$ mV and $V_{\text{store}} = 350$ mV. For all following experiments shown in this study ambient temperature was kept at 25 °C. The data shown in Fig. 15 is corrected for different offsets of the readout on different chips. All curves are shifted vertically, so that without stimulation the average $\langle a_\pm \rangle$ lies at 1.00 V. This way the curves can be shown and compared in one plot. For learning applications the offset is determined on program startup by the PPU.

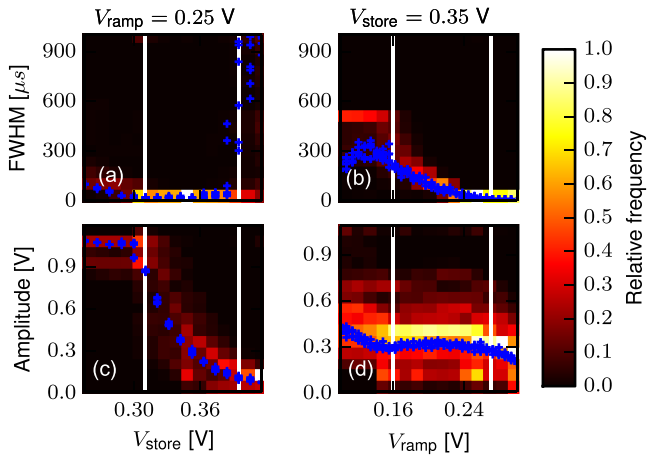

Fig. 16. Width and amplitude of the STDP interaction box in dependence of parameters V_{store} (left column) and V_{ramp} (right column). The blue crosses show data for one single synapse. White bars mark the useful range for the corresponding parameter (see text). Colors indicate the relative frequency for a total of 192 synapses on three different chips.

The results show biologically realistic time-constants of approximately 20 μs to be achievable. Here, we use a speed-up factor of 10^3 to convert from biological time-constants of approximately 20 ms given in [39]–[41]. The average time-constants in Fig. 15 are $\langle\hat{\tau}_{\pm}\rangle = 30$ μs with a standard deviation of 10 μs for Fig. 15(a) and (b) and 8 μs for Fig. 15(c) and (d). The achievable ranges are discussed later (see Fig. 16).

Trial-to-trial variability for individual synapses is generally small. The mean trial-to-trial standard deviation for all four plots is equal within errors at 8 ± 5 mV. Therefore, the variation between synapses that can be seen in the plots is due to device mismatch within the synapse circuit itself and mismatch within the readout channels of the ADC. Plots C and D of Fig. 15 show only data for a single channel each. Concerning amplitude, standard deviations for the multi- and single-channel cases are comparable: $\langle\hat{a}_{\pm}\rangle = 400 \pm 140$ mV for A and C, $\langle\hat{a}_{\pm}\rangle = 600 \pm 180$ mV for B and D. For the time-constants single-channel data exhibit slighlty less variability [see Fig. 15(c) and (d)]. However, differences are small and overall variability can be assumed to be dominated by mismatch between the synapse circuits themselves.

C. Achievable Ranges

To configure the shape of the STDP curve the circuit provides two primary configuration parameters: V_{store} and V_{ramp} (see Section II-A). V_{store} controls the storage gain and V_{ramp} the time constant (see Fig. 5). We measured 192 synapses on three different chips sweeping both parameters to find the achievable amplitudes $\hat{a}_{\pm}$ and widths $\hat{\tau}_{\pm}$. Fig. 16 shows the results while using $N = 32$ spike pairs (see Section III-A for the definiton of the plotted quantities "width" and "amplitude").

The usable range is the parameter range, for which V_{store} controls amplitude and V_{ramp} controls the width. The respective other property, i.e., width for V_{store} and amplitude for V_{ramp}, remains flat. Therefore, the shape of the STDP curve can be tuned with the given parameters. For the presented measurements the usable ranges were selected as $V_{\text{store}} \in [0.31 \ldots 0.40$ V] and $V_{\text{ramp}} \in [0.16 \ldots 0.27$ V]. This range lies between the white vertical markers in Fig. 16. Table IV gives mean and standard deviation at start and stop of this range for $\hat{a}$ and $\hat{\tau}$. The amplitude covers nearly the 1 V of full dynamic range of the ADC input. Time-constants show a large configurable range from tens to hundreds of micro-seconds. Even lower values down to 2 μs are configurable, but the error will stay at 4 μs so that we have excluded these values from the usable range. The amplitude can maximally be as large as the available input range of the ADC, which is evident in the measured data.

TABLE IV
ACHIEVABLE RANGES

Parameter	Start	Stop	Unit
$\hat{a}_+$	0.91 ± 0.05	0.20 ± 0.06	V
$\hat{\tau}_+$	336.60 ± 108.43	11.36 ± 3.08	μs
$\hat{a}_-$	0.91 ± 0.14	0.15 ± 0.04	V
$\hat{\tau}_-$	228.73 ± 85.30	12.31 ± 3.57	μs

D. Full-System Experiments

With the individual channels characterized, the next step is to look at the full signal processing chain. We use the experimental protocol described in Section III and illustrated in Fig. 12. The PPU performs weight updates according to (18). To eliminate trial-to-trial noise on the analog readout and to remove systematic offset between the two channels of one synapse, we modify (17) to

$$\bar{A} = (A_+ - A_-) - A_{\text{off}} \tag{20}$$

$$A = \begin{cases} \bar{A} & \text{if } |\bar{A}| > \theta \\ 0 & \text{else.} \end{cases} \tag{21}$$

Here, A_{off} is determined at program startup after reset of the accumulation storage as difference $A_+ - A_-$. (21) implements thresholding using the user selected parameter θ. The PPU performs updates at regular intervals of 10 μs during stimulation. The source code for the actually used update program is available from [42].

Fig. 17 shows results when using 8 bit resolution for arithmetics. For analysis, two functions f_+ and f_- are individually fitted to pre-before-post ($\Delta t > 0$) and post-before-pre ($\Delta t < 0$) data

$$f_+(\Delta t) = w + b_+(w_{\max} - w)\exp\left(-\frac{\Delta t}{c_+}\right) \tag{22}$$

$$f_-(\Delta t) = w - wb_-\exp\left(\frac{\Delta t}{c_-}\right). \tag{23}$$

Here, $b_{\pm}$ and $c_{\pm}$ are the fit parameters, while initial weight w and maximum weight $w_{\max}$ are the same as those used by the update program. In all experiments we set $w_{\max}$, λ, and α to 1.0. The threshold θ was set to 10. Since discretization of the weight removes the long tail of the exponential, the fit is restricted to points where the weight was actually changed ($w' \neq w$).

Fig. 17(a)–(c) demonstrate different combinations of N and w. Especially for small updates, the discretization of the weight to 6 bit is apparent. Results exhibit the expected dependency on w for a multiplicative rule. Fig. 17(d) and (e) plot the fitted parameters for amplitude ($b_{\pm}$) and time-constant ($c_{\pm}$) over the

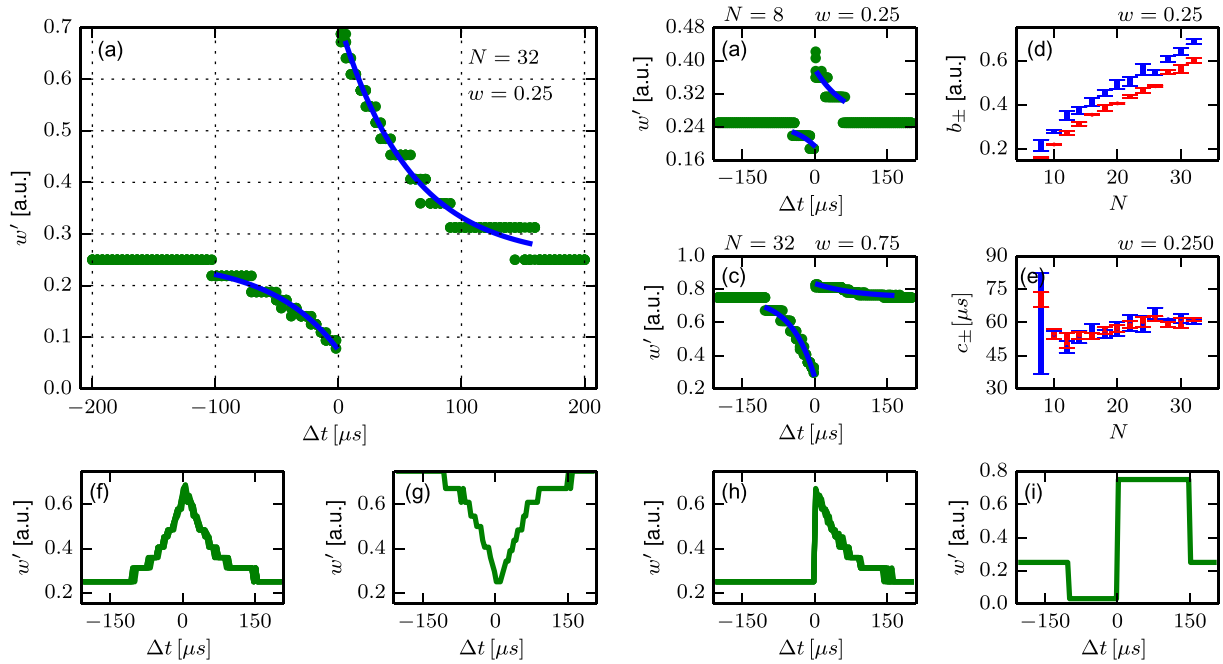

Fig. 17. Results from experiments using the full signal chain including PPU, ADC, and correlation sensor in the synapse. Synapses are stimulated according to the protocol outlined in Fig. 12. The PPU computes a multiplicative update rule according to (21) and (18). (a)–(c) Weight after stimulation for five repetitions as green points. Fit to data as blue curve. (d) and (e) Resulting fit parameters for amplitude b_+ (red), b_- (blue) and time-constant c_+ (red), c_- (blue) for multiple number of spike-pairs N. (f)-(i) Examples of other updating rules that can be implemented.

number of spike pairs N. As expected, amplitude increases linearly with the number of pairs and for the chosen initial weight $w = 0.25$ positive changes are larger than negative ones. The process that measures the timing of spike pairs in the synapse operates on individual pairs and is therefore independent of N. Also, the circuit for time measurement is shared for pre-before-post and post-before-pre pairs within one synapse. Therefore, time-constants should be identical on both sides and independent of N. Experimental data is compatible with these expectations as Fig. 17(e) shows. For small N the fit is not reliable due to discretization [see Fig. 17(b)].

The plots in Fig. 17(f)–(i) give a hint of the achievable flexibility. They were produced with the same stimulation protocol only by changes in software running on the PPU. Fig. 17(f) and (g) show symmetrical Hebbian and anti-Hebbian rules. Fig. 17(h) is only sensitive to pre-before-post pairings. Fig. 17(i) realizes bi-stable learning.

E. Power Consumption

During execution of the experiment described in the previous section, digital logic consumes below 32 mW of power as measured on the power supply pins of the chip. With the clock disabled for the PPU, power consumption drops below 10 mW, so that 22 mW can be attributed to the PPU. In reset, power consumption drops by 2 mW for the PPU. Therefore, power consumption is largely due to clock distribution.

VI. Discussion of Results

The two overarching goals in the development of neuromorphic hardware are to provide a platform for neuroscientific experiments and to find new ways of computation for technical applications. For both these goals we believe reliability, scalability, and flexibility to be enabling factors besides efficiency in terms of power, area, and speed. Therefore, the presented results focus on these aspects.

A. Reliability

To assess reliability we characterized the synapse behavior across three different chips (see Figs. 15 and 16). Results show substantial variation due to device mismatch within the analog circuits. Please note, that for these measurements the configuration bits of the synapse circuit were not even used (see Section II-A). So there is room for improvements through calibration. On the other hand, trial-to-trial variability of individual components is small. This is for example illustrated in Fig. 16 that shows multiple trials from a single synapse on the background of the overall distribution of synapses. A small trial-to-trial variability was also measured for individual channels in Section V-B. This allows on the one hand to use off-line calibration, but on the other hand it is also conceivable, that an emulated network calibrates itself through the use of plasticity. Indeed, the robustness of reward-based learning to device mismatch on the correlation detection within the processor-based approach presented here has been shown in previous work [23]. Self-tuning has also been shown to be feasible through the use of short-term plasticity [43].

In general, a plasticity mechanism can compensate inhomogeneities if there is a feedback loop for the parameter subject to variation. This is typically the case for outputs, e.g., synaptic weights. An STDP rule will modify the weight according to the timing behavior it observes, which is an effect of the weight including variation. Variation on the input however—in this

case the signal $a_{\pm}$ from the correlation sensor—is invisible from the rule. It can however be compensated by introducing additional information about the behavior of the system, for example through a reward signal. This addition of complementary information is why reward-based learning rules are well suited for analog neuromorphic hardware systems. The alternative is to use redundancy of analog components so that the average behavior is reliable.

B. Scalability

Scalability can of course only be shown by actually scaling the system, which we plan to do in the future. Nonetheless, the plasticity system is designed to scale well: the only part for which area scales linearly with the number of synapses is the correlation sensor that resides within the synapse circuit. Therefore, we have chosen to use an area-optimized circuit realized as analog full-custom design. The ADC scales with the number of columns in the synapse array, which have typically a square root dependency on the number of synapses. Most parts of the PPU are required only once per array and only the number of vector slices scales with the number of columns. To keep these slices as lightweight as possible, all control logic is shared and a single-port vector register file is used. A scaled system will feature arrays of 256×256 synapses with a dedicated PPU using 8 vector slices.

C. Flexibility

The whole approach presented here has a strong emphasis on flexibility, compared to our previous implementation [14] and considering the constraints of an analog, accelerated neuromorphic system. By this we mean, that a large number of plasticity rules should be implementable in the hardware system. Introducing the PPU sacrifices area and power in order to have as much freedom as possible while not sacrificing speed. To achieve this latter point in the 65 nm technology, we consider a combination of analog and software-based processing, as shown in this study, to be necessary. At a speed-up factor of 10^3 and array sizes of 65 k synapses it is not feasible to process individual spike events in software. This of course limits flexibility as the functionality of the correlation sensor is fixed in hardware. Therefore, this functionality should at least operate over a wide range of parameters, demonstrated by the results shown in Fig. 16 and Table IV In the biological time domain, the design covers ranges from tens to hundreds of milliseconds, fitting typical ranges found in biology [39]–[41]. Also the amplitude is tunable over a large range, so that the sensitivity of the correlation sensor can be matched to the network activity.

In general, every plasticity model is implementable in this system that depends only on observables visible to the PPU and affects only parameters accessible by the PPU. Observables are the weight w, the correlation signals $a_{\pm}$, a firing rate sensor not discussed in this study, and signals from outside the chip such as reward. All parameters of the chip that can be modified at all, can also be modified by the PPU. This includes the synaptic weight w, neuron parameters, and the topology of the network. The latter is limited to the addresses stored in the synapses for this prototype chip.

In future realizations it is feasible to increase the number of observables of the PPU. It is planned to include a fast ADC in a forthcoming chip which will give the PPU access to membrane voltages. It is also feasible to add synapse correlation measurement circuits with novel properties, if there are plasticity models demanding them.

Here we only show the simple STDP rule given in (14) as proof of concept. Fig. 17(f)–(i) show simple examples of modifications of the plasticity model purely realized in software running on the PPU. Beyond that, the reward-based learning rule R-STDP and a learning rule for spike-based expectation maximization has been ported to the system, but not yet tested in hardware [23], [44].

VII. Conclusion

In this study we have presented a new approach to plasticity in neuromorphic hardware: the combination of dedicated analog circuits in every synapse with a shared digital processor. It represents a trade-off between flexibility of implementable plasticity models and efficiency of the implementation in terms of area, speed, and energy. The presented results demonstrate the viability of this approach for plasticity.

The more classical approach taken for neuromorphic hardware, for example by [10] or [45], is to implement a single plasticity mechanism that can be used to solve a range of network learning tasks. Analog continuous-time implementations of neuromorphic circuits can be combined with floating-gate technology to achieve persistence of the learned synaptic weights. By modifing the control signals the precise learning rules can also be tuned [46]. Our approach not only aims for flexibility in the learning task, but also in the mechanism itself. Together with the speed-up factor this enables experimental analysis of long-term effects of such mechanisms. In the classical approach it is essential to have a detailed understanding of the mechanism prior to production of hardware. In our approach the hardware system can help to gain this understanding. This is an important aspect when designing a system intended as a neuroscientific platform.

In [47] this approach is taken even further: neuronal dynamics as well as detection of correlations and weight update are performed by general-purpose processors in software. Specialized hardware is only used for event communication. This maximizes flexibility but further sacrifices efficiency, so that operation is only possible without speed-up. Another mixed-mode approach is reported in [48]. Here, the authors also perform the full plasticity operation in software, achieving maximum flexibility, while the synapses and neurons are full-custom analog implemetations.

Our approach to use dedicated hardware for the most expensive part—the processing of spikes—enables faster operation. Using an on-die PPU local to the synapse circuits also facilitates scaling of the system, since no communication to off-chip components is necessary. Since learning and development in biology are processes spanning many time-scales, platforms for accelerated simulation or emulation are important. In the

domain of general-purpose computers using software simulations even for medium-sized networks accelerated operation with plasticity is currently not possible [21].

VIII. Outlook

The chip presented here is still an early prototype that for example lacks on-chip networking capabilities. However, using the experimental setup described here a wide range of plasticity mechanisms can already be implemented and analyzed in hardware. Obvious candidates are the models already prepared for implementation [23], [44]. Future prototypes will add the ability to include neuronal and structural plasticity opening the door for a large set of learning mechanisms. It will then also be possible to execute learning tasks involving networks of neurons with the system.

In the long run, the focus will be on scaling the system in size. As an intermediate step we plan to build chip-scale variants with two 256×256 synapse arrays and two PPUs. Eventually, the goal is to go to wafer-scale [7]. It will then replace the first generation of the neuromorphic platform (NM-PM-1) of the Human Brain Project [49].

We also hope that the release of the PPU design—the Nux processor [28]—as open source will turn out to be a valuable contribution to open source hardware.

References

[1] L. Yann, B. Yoshua, and H. Geoffrey, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Comput.*, vol. 4, no. 2, pp. 234–242, 1992.

[3] G. E. Hinton, "Learning multiple layers of representation," *Trends Cognitive Sci.*, vol. 11, no. 10, pp. 428–434, 2007.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Red Hook, NY, USA: Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[5] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[6] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E. S. Chung, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Research Whitepaper*, vol. 2, 2015.

[7] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2010, pp. 1947–1950.

[8] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12pp. 2454–2467, 2012.

[9] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[10] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A re-configurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Front. Neurosci.*, vol. 9, no. 141, 2015.

[11] C. A. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, pp. 1629–1636, 1990.

[12] R. Douglas, M. Mahowald, and C. Mead, "Neuromorphic analogue VLSI," *Annu. Rev. Neurosci.*, vol. 18, pp. 255–281, 1995.

[13] G. Indiveri, "A low-power adaptive integrate-and-fire neuron circuit," in *Proc. ISCAS*, 2003, vol. 4, pp. 820–823.

[14] J. Schemmel, A. Grübl, K. Meier, and E. Muller, "Implementing synaptic plasticity in a VLSI spiking neural network model," in *Proc. Int. Joint Conf. Neural Networks*, 2006.

[15] J. H. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour," *Neural Netw.*, vol. 21, no. 2/3, pp. 524–534, 2008.

[16] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: taming the beast," *Nature Neurosci.*, vol. 3, pp. 1178–1183, 2000.

[17] N. Caporale and Y. Dan, "Spike timing-dependent plasticity: A hebbian learning rule," *Annu. Rev. Neurosci.*, Feb. 2008. doi: http://dx.doi.org/10.1146/annurev.neuro.31.060407.125639.

[18] R. Ananthanarayanan, S. Esser, H. Simon, and D. Modha, "The cat is out of the bag: Cortical simulations with 10 9 neurons, 10 13 synapses," in *Proc. Conf. High Perform. Comput. Netw. Storage. Anal.*, 2009, p. 63.

[19] M. Helias, S. Kunkel, G. Masumoto, J. Igarashi, J. M. Eppler, S. Ishii, T. Fukai, A. Morrison, and M. Diesmann, "Supercomputers ready for use as discovery machines for neuroscience," *Front. Neuroinform.*, vol. 6, no. 26, 2012, doi: 10.3389/fninf.2012.00026.

[20] H. Markram, E. Muller, S. Ramaswamy, M. Reimann, M. Abdellah, C. Sanchez, A. Ailamaki, L. Alonso-Nanclares, N. Antille, S. Arsever, G. Kahou, T. Berger, A. Bilgili, N. Buncic, A. Chalimourda, G. Chindemi, J.-D. Courcol, F. Delalondre, V. Delattre, S. Druckmann, R. Dumusc, J. Dynes, S. Eilemann, E. Gal, M. Gevaert, J.-P. Ghobril, A. Gidon, J. Graham, A. Gupta, V. Haenel, E. Hay, T. Heinis, J. Hernando, M. Hines, L. Kanari, D. Keller, J. Kenyon, G. Khazen, Y. Kim, J. King, Z. Kisvarday, P. Kumbhar, S. Lasserre, J.-V. Le Bé, B. Magalhães, A. Merchán-Pérez, J. Meystre, B. Morrice, J. Muller, A. Muñoz-Céspedes, S. Muralidhar, K. Muthurasa, D. Nachbaur, T. Newton, M. Nolte, A. Ovcharenko, J. Palacios, L. Pastor, R. Perin, R. Ranjan, I. Riachi, J.-R. Rodríguez, J. Riquelme, C. Rössert, K. Sfyrakis, Y. Shi, J. Shillcock, G. Silberberg, R. Silva, F. Tauheed, M. Telefont, M. Toledo-Rodriguez, T. Tränkler, W. Van Geit, J. Díaz, R. Walker, Y. Wang, S. Zaninetta, J. DeFelipe, S. Hill, I. Segev, and F. Schürmann, "Reconstruction and simulation of neocortical microcircuitry," *Cell*, vol. 163, no. 2, pp. 456–492, 2015.

[21] F. Zenke and W. Gerstner, "Limits to high-speed simulations of spiking neural networks using general-purpose computers," *Front. Neuroinform.*, vol. 8, no. 76, 2014, doi: 10.3389/fninf.2014.00076.

[22] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *Proc. 11th USENIX Symp. Operating Syst. Des. Implementation*, 2014, pp. 571–582.

[23] S. Friedmann, N. Frémaux, J. Schemmel, W. Gerstner, and K. Meier, "Reward-based learning under hardware constraints—Using a RISC processor embedded in a neuromorphic substrate," *Front. Neurosci.*, vol. 7, p. 160, 2013. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2013.00160

[24] J. Schemmel, D. Brüderle, K. Meier, and B. Ostendorf, "Modeling synaptic plasticity within networks of highly accelerated I& F neurons," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2007, pp. 3367–3370.

[25] Y. Dan and M.-M. Poo, "Spike timing-dependent plasticity: From synapse to perception," *Phys. Rev.*, vol. 86, no. 3, pp. 1033–1048, 2006.

[26] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biol. Cybern.*, vol. 98, no. 6, pp. 459–478, Jun. 2008. .

[27] PowerISA, "PowerISA Version 2.06 Revision b," Tech. Rep., Jul. 2010, [Online]. Available: http://www.power.org/resources/reading/

[28] S. Friedmann, "The Nux Processor v3.0," 2015. DOI: 10.5281/zenodo.32146. [Online]. Available: https://github.com/electronicvisions/nux

[29] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic aps," *Science*, vol. 275, pp. 213–215, 1997.

[30] P. J. Sjöström, G. G. Turrigiano, and S. B. Nelson, "Rate, timing, and cooperativity jointly determine cortical synaptic plasticity," *Neuron*, vol. 32, no. 6, pp. 1149–1164, 2001.

[31] P. J. Sjöström, G. G. Turrigiano, and S. B. Nelson, "Endocannabinoid-dependent neocortical layer-5 ltd in the absence of postsynaptic spiking," *J. Neurophys.*, vol. 92, no. 6, pp. 3338–3343, 2004.

[32] R. C. Froemke, D. Debanne, and G.-Q. Bi, "Temporal modulation of spike-timing-dependent plasticity," *Front. Synap. Neurosci.*, vol. 2, no. 19, 2010.

[33] R. C. Froemke, J. J. Letzkus, B. Kampa, G. B. Hang, and G. Stuart, "Dendritic synapse location and neocortical spike-timing-dependent plasticity," *Front. Synap. Neurosci.*, vol. 2, no. 29, 2010.

[34] D. Liu, " Chapter 2—Numerical representation and finite-length {DSP}," in *Embedded {DSP} Processor Design,* ser. Systems on Silicon, D. Liu, Ed. Burlington: Morgan Kaufmann, 2008, vol. 2, pp. 47–86.

[35] T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier, "Is a 4-bit synaptic weight resolution enough?—Constraints on enabling spike-timing dependent plasticity in neuromorphic hardware," *Front. Neurosci.*, vol. 6, no. 90, 2012.
[36] D. S. Greenberg, A. R. Houweling, and J. N. D. Kerr, "Population imaging of ongoing neuronal activity in the visual cortex of awake rats," *Nature Neurosci.*, vol. 11, pp. 749–751, 2008.
[37] C. P. de Kock and B. Sakmann, "Spiking in primary somatosensory cortex during natural whisking in awake head-restrained rats is cell-type specific," *Proc. Nat. Acad. Sci.*, vol. 38, pp. 16 446–16 450, 2009.
[38] T. C. Potjans and M. Diesmann, "The cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network modela," *Cereb. Cortex*, vol. 24, pp. 785–806, 2012.
[39] H. Markram and B. Sakmann, "Action potentials propagating back into dendrites trigger changes in efficacy of single-axon synapses between layer v pyramidal neurons," in *Proc. Soc. Neurosci. Abstr.*, 1995, vol. 21, p. 2007.
[40] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neuroscience, Official J. Soc. Neurosci.*, vol. 18, no. 24, pp. 10 464–10 472, Dec. 1998.
[41] Y. Dan and M. Poo, "Spike timing-dependent plasticity of neural circuits," *Neuron*, vol. 44, no. 1, pp. 23–30, Sep. 2004.
[42] S. Friedmann, " PPU Software v1.0," 2015. doi: 10.5281/zenodo.32147. [Online]. Available: https://github.com/electronicvisions/ppu-software
[43] J. Bill, K. Schuch, D. Brüderle, J. Schemmel, W. Maass, and K. Meier, "Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity," *Front. Comp. Neurosci.*, vol. 4, no. 129, 2010.
[44] O. Breitwieser, "Towards a Neuromorphic Implementation of Spike-Based Expectation Maximization," Master thesis, Kirchhoff Institute for Physics, Ruprecht-Karls-Universität Heidelberg, 2015.
[45] S. Ramakrishnan, P. Hasler, and C. Gordon, "Floating gate synapses with spike-time-dependent plasticity," *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 3, pp. 244–252, 2011.
[46] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 1, pp. 71–81, Feb. 2013.
[47] F. Galluppi, X. Lagorce, E. Stromatias, M. Pfeiffer, L. A. Plana, S. B. Furber, and R. B. Benosman, "A framework for plasticity implementation on the SpiNNaker neural architecture," *Front. Neurosci.*, vol. 8, no. 429, 2015.
[48] M. R. Azghadi, S. Moradi, D. B. Fasnacht, M. S. Ozdas, and G. Indiveri, "Programmable spike-timing-dependent plasticity learning circuits in neuromorphic VLSI architectures," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 2, pp. 17:1–17:18, Sep. 2015.
[49] H. Markram, K. Meier, T. Lippert, S. Grillner, R. Frackowiak, S. Dehaene, A. Knoll, H. Sompolinsky, K. Verstreken, J. DeFelipe, S. Grant, J.-P. Changeux, and A. Saria, "Introducing the human brain project," *Procedia Comput. Sci.*, vol. 7, pp. 39–42, 2011.

Simon Friedmann received the Dipl. Phys. and Ph.D. degrees from Heidelberg University, Heidelberg, Germany in 2009 and 2013, respectively.

Currently, he is a Postdoctoral Researcher in the Electronic Vision(s) group at Heidelberg University. His research focus is on learning neuromorphic hardware.

Johannes Schemmel (M'08) received the Ph.D. degree in physics from Heidelberg University, Heidelberg, Germany, in 1999.

Currently, he is 'Akademischer Oberrat' in the Kirchhoff Institute of Physics, Heidelberg, where he is head of the ASIC laboratory and the Electronic Vision(s) group. His research interests are mixed-mode VLSI systems for information processing, especially the analog implementation of biologically realistic neural network models. He is the architect of the Spikey and BrainScaleS accelerated Neuromorphic hardware systems.

Andreas Grübl received the Dipl. Phys. and Ph.D. degrees from Heidelberg University, Heidelberg, Germany in 2003 and 2007, respectively.

Currently, he is a Senior Postdoctoral Researcher in the Electronic Vision(s) group and leader of the Electronics Department of the Kirchhoff Institute for Physics at Heidelberg University. He has eight years of postdoctoral experience in designing and building complex microelectronics systems for brain-inspired information processing. His research focus is on new methods for the implementation of large mixed-signal SoCs.

Andreas Hartel received the Dipl. Phys. and Ph.D. degrees from Heidelberg University, Heidelberg, Germany in 2010 and 2016, respectively.

Currently, he is a Postdoctoral Researcher in the Electronic Vision(s) group at Heidelberg University. His research focus is on learning in neuromorphic hardware.

Matthias Hock received the Dipl. Phys. and Ph.D. degrees from Heidelberg University, Heidelberg, Germany in 2009 and 2015, respectively.

Currently, he is a Postdoctoral Researcher in the Electronic Vision(s) group at Heidelberg University. His research focus is on development and test of mixed-signal circuits for neuromorphic hardware.

Karlheinz Meier received the Ph.D. degree in physics from Hamburg University, Hamburg, Germany, in 1984.

Currently, he is a Professor of Physics at Heidelberg University, Heidelberg, Germany, and Co-Founder of the Kirchhoff-Institut and the Heidelberg ASIC Laboratory in Heidelberg. His research interests include the application of microelectronics in particle physics, electronic realizations of brain circuits, and principles of information processing in spiking neural networks.