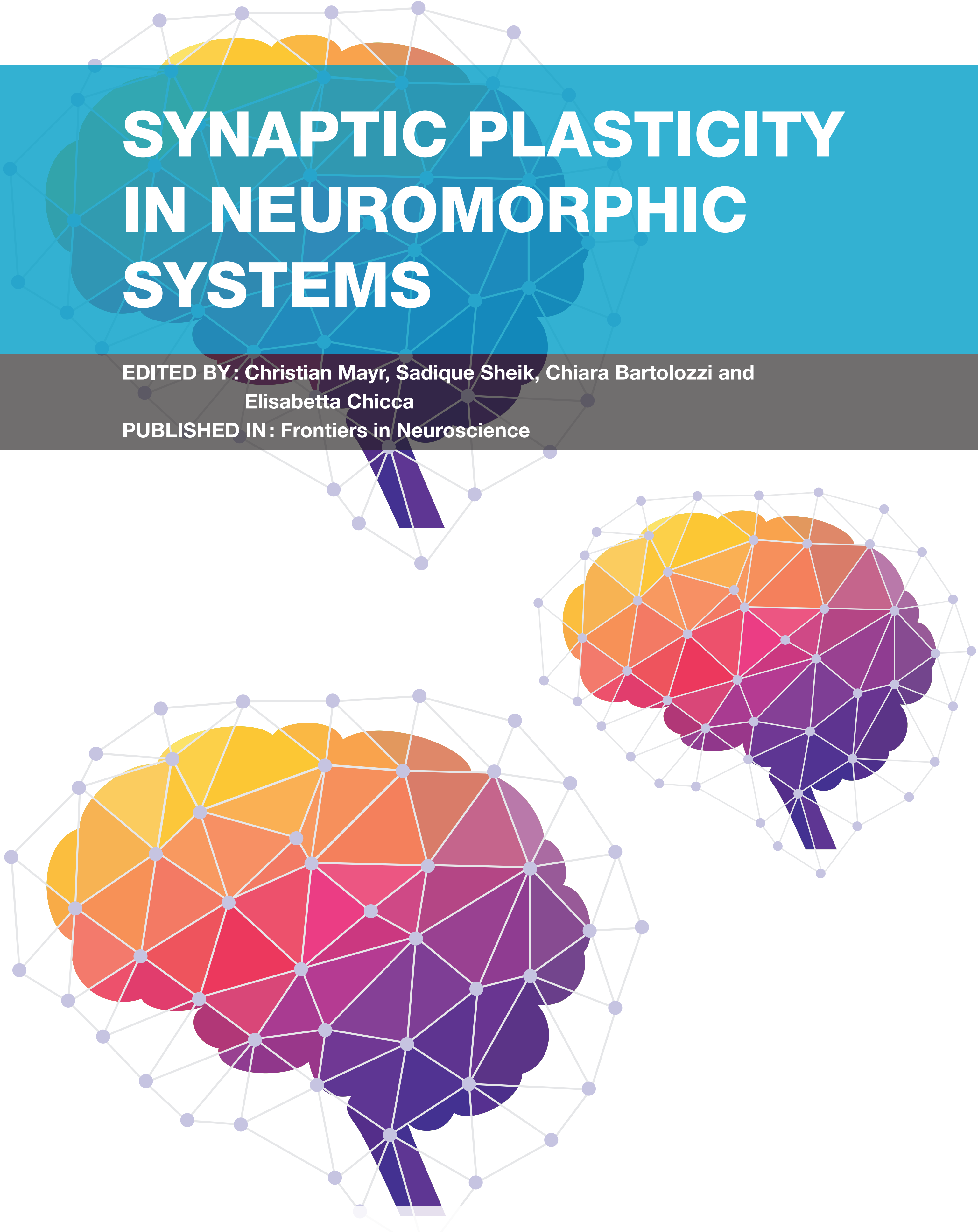
SYNAPTIC PLASTICITY IN NEUROMORPHIC SYSTEMS

EDITED BY: Christian Mayr, Sadique Sheik, Chiara Bartolozzi and Elisabetta Chicca

PUBLISHED IN: Frontiers in Neuroscience



frontiers Research Topics

ISSN 1664-8714
ISBN 978-2-88919-877-1
DOI 10.3389/978-2-88919-877-1

SYNAPTIC PLASTICITY IN NEUROMORPHIC SYSTEMS

Topic Editors:
Christian Mayr, Technische Universität Dresden, Germany
Sadique Sheik, University of California, San Diego, USA
Chiara Bartolozzi, Istituto Italiano di Tecnologia, Italy
Elisabetta Chicca, Bielefeld University, Germany

One of the most striking properties of biological systems is their ability to learn and adapt to ever changing environmental conditions, tasks and stimuli. It emerges from a number of different forms of plasticity, that change the properties of the computing substrate, mainly acting on the modification of the strength of synaptic connections that gate the flow of information across neurons.

Plasticity is an essential ingredient for building artificial autonomous cognitive agents that can learn to reliably and meaningfully interact with the real world. For this reason, the neuromorphic community at large has put substantial effort in the design of different forms of plasticity and in putting them to practical use. These plasticity forms comprise, among others, Short Term Depression and Facilitation, Homeostasis, Spike Frequency Adaptation and diverse forms of Hebbian learning (e.g. Spike Timing Dependent Plasticity).

This special research topic collects the most advanced developments in the design of the diverse forms of plasticity, from the single circuit to the system level, as well as their exploitation in the implementation of cognitive systems.

Citation: Mayr, C., Sheik, S., Bartolozzi, C., Chicca, E., eds. (2016). Synaptic Plasticity in Neuromorphic Systems. Lausanne: Frontiers Media. doi: 10.3389/978-2-88919-877-1

Table of Contents

Editorial: Synaptic Plasticity for Neuromorphic Systems

Christian G. Mayr[1]*, *Sadique Sheik*[2], *Chiara Bartolozzi*[3] *and Elisabetta Chicca*[4]

[1] *Chair of Highly-Parallel VLSI-Systems and Neuromorphic Circuits, Technische Universität Dresden, Dresden, Germany,* [2] *BioCircuits Institute, University of California, San Diego, San Diego, CA, USA,* [3] *iCub Facility, Istituto Italiano di Tecnologia, Genova, Italy,* [4] *Cognitive Interaction Technology - Center of Excellence, Faculty of Technology, Bielefeld University, Bielefeld, Germany*

Keywords: synaptic plasticity, neuromorphic engineering, memristive plasticity, plasticity circuits, digital plasticity, high-density plasticity, plasticity for sensor data, learning feature extraction

The Editorial on the Research Topic

Synaptic Plasticity for Neuromorphic Systems

Brain plasticity serves animals in a wide range of vital functions. It assists them in adapting their behavior to the surroundings, in learning new strategies for optimizing a certain reward-seeking policy for their survival or in adjusting motor activity through sensory feedback. Thus, plasticity is an essential ingredient for building artificial autonomous systems that can cope with the real world. In order to build these systems, neuromorphic design labs actively investigate and develop various circuit implementations of plasticity. This research topic collects a comprehensive snapshot of this work. A number of manuscripts published in this topic study the interplay between stochasticity and plasticity (Afshar et al.; Bill and Legenstein; Lagorce et al.; Qiao et al.). Plasticity here acts in a stochastic fashion or extracts features from stochastic sensor data. The current push toward higher complexity/scale in neuromorphic devices can also be observed in plasticity implementations (Qiao et al.; Wang et al.; Noack et al.). Due to advantageous technology scaling and reproducibility, digital implementations of neuromorphic plasticity are gaining popularity (Galluppi et al.; Vogginger et al.). The collection of articles in this topic is rounded out by articles on plasticity in novel nano-scale technologies (Saighi et al.; Thomas et al.; Wang et al.; Bill and Legenstein).

OPEN ACCESS

Edited by:
Timothy K. Horiuchi, The University of Maryland, USA

Reviewed by:
Theodore Yu, Texas Instruments Inc., USA

Correspondence:
Christian Mayr
christian.mayr@tu-dresden.de

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *27 January 2016*
Accepted: *27 April 2016*
Published: *19 May 2016*

Citation:
Mayr CG, Sheik S, Bartolozzi C and Chicca E (2016) Editorial: Synaptic Plasticity for Neuromorphic Systems. Front. Neurosci. 10:214. doi: 10.3389/fnins.2016.00214

1. STOCHASTICITY AND PLASTICITY

One topic of interest in recent publications is the interaction between stochasticity and synaptic dynamics. Wang et al. introduces a stochastic synapse cell constructed with a memristor and two transistors. Bill and Legenstein show that stochastic synapses can provide graded responses from binary-valued synapses, aiding convergence in learning tasks. Stochasticity in conjunction with plasticity can also aid error tolerance. For instance, the stochastic synapse model of Bill and Legenstein can learn handwritten digits with high fidelity in the presence of significant device deviations and noise. The statistical inference in Afshar et al. actually uses deviations between individual dendrites. The visual feature extraction in Lagorce et al. operates on a high-dimensional projection of the input space through a recurrent neural network, benefitting from deviations across elements. A more conventional error-compensation approach is taken in Qiao et al., where a network counterbalances for deviations through a learned aggregate of individual neuronal responses.

2. PLASTICITY OPERATING ON SENSOR DATA

The above mentioned Qiao et al. and Lagorce et al. also represent examples of processing and plasticity operating directly on spiking input. In fact, typical tasks that would be amenable to a neuromorphic solution have traditionally used non-spiking input, such as image processing applications exclusively using image frames (Henker et al., 2007). Due to these incompatible representations (e.g., continuous time vs. discrete time, spikes vs. scalar values), there has not been much synergy between neuromorphic and traditional sensor processing, thus potentially missing some novel approaches in both fields. However, the two are growing closer together as sensors with spiking output are becoming more widely available in such diverse areas as vision (Delbruck, 2008) or audition (Liu et al., 2010). In addition to the plastic sensory processing in Qiao et al. and Lagorce et al., the statistical inference of Afshar et al. could also be employed for sensory processing, as it is geared toward the temporal patterns of multiple input spike trains.

3. DIGITAL IMPLEMENTATIONS OF PLASTICITY

Neuromorphic engineering was envisioned as analog VLSI circuits, due to the similarity between the current across CMOS devices in subthreshold and across neurons ion channels. However, digital circuits benefit significantly more from technology scaling and low-power advances in deep-submicron nodes, making them attractive for neuromorphic implementations. Specifically, plasticity allows fixed, reproducible-function digital circuits to add adaptability and variation to their behavior. Galluppi et al. present a framework for plasticity implementation on a programmable digital neuromorphic system, SpiNNaker. Vogginger et al. discuss a computational optimization of a powerful learning rule, outlining an efficient implementation in a synthesized or programmable digital neuromorphic system. Afshar et al. present an FPGA implementation of a novel neuron model and an accompanying learning rule optimized for digital circuits.

4. LARGE SCALE HARDWARE FOR PLASTICITY

Complex real-world applications demand large, computationally capable neural networks. Consequently, there is a drive toward large scale neuromorphic hardware with plasticity. The chip of Qiao et al. is currently one of the largest devices with on-chip plasticity (256 neurons, 128k synapses, 180 nm CMOS) that employs the original subthreshold design philosophy. Noack et al. present a switched capacitor implementation of short- and long-term plasticity in 28 nm CMOS that at $3.6 \times 3.6\ \mu m^2$ is an order of magnitude smaller than any other plastic CMOS synapse. Other approaches to scaling include Wang et al., which uses a digital time-multiplexed circuit to compute Spike Timing Dependent Plasticity (STDP) for time-multiplexed analog neurons. For large-scale networks, topological considerations also play an increasing role, e.g., in terms of which signals a plasticity circuit needs access to (e.g., pre- or post-synaptic) (Noack et al., 2010). A neuron-synapse matrix arrangement seems the obvious choice, but the implementations in this topic explore a variety of other options.

5. MEMRISTIVE PLASTICITY

In terms of emerging technologies, the usage of nanoscale memristors for short- or long term plasticity has seen a large deal of interest since the pioneering work of Jo et al. (2010). Memristors inherently replicate aspects of synaptic plasticity and can combine plasticity, weight storage and weight effect in a single device. Saighi et al. gives an overview of recent developments in this area from a materials and neuromorphic perspective. Thomas et al. investigate tunnel junction based memristors that exhibit STDP-like plasticity. Wang et al. present a synaptic cell composed of memristor plus transistors which endows the synapse with stochastic learning capabilities. Bill and Legenstein introduce a model of an ideal stochastic memristor synapse and investigate its computational properties.

6. SUMMARY

Synaptic plasticity is a crucial ingredient in neuromorphic hardware. It has the potential to contribute to many different fields, such as in the endeavor of building realistic brain models, in biohybrids where the hardware adapts to the biological counterpart or in the construction of truly cognitive systems. This research topic gives an overview of the state-of the art in plasticity circuit design and applications and outlines future research directions.

AUTHOR CONTRIBUTIONS

CM, EC, CB, and SS all contributed to the compilation of plasticity works described in this editorial and to the writing of the editorial.

ACKNOWLEDGMENTS

This work was supported by the Excellence Cluster 227 (CITEC, Bielefeld University), the U.S. Office of Naval Research(ONR) under Grant Number N00014-13-1-0205 and the Swiss National Science Foundation under grant number P2EZP2_155561. This research has received funding from the European Union Seventh Framework Programme (FP7/2007- 2013) under grant agreement no. 269459 (CORONET), no. 612058 (RAMP) and no. 284553 (SICODE). The authors would like to thank all contributors of the Frontiers Special Topic. We wish to acknowledge R. Douglas, G. Indiveri, S. Fusi and F. Stefanini for insightful discussions.

REFERENCES

Delbruck, T. (2008). "Frame-free dynamic digital vision," in *Proceedings of the International Symposium on Secure-Life Electronics*, Vol. 1 (Tokyo: University of Tokyo).

Henker, S., Mayr, C., Schlüssler, J.-U., Schüffny, R., Ramacher, U., and Heittmann, A. (2007). "Active pixel sensor arrays in 90/65nm CMOS-technologies with vertically stacked photodiodes," in *Proceedings IEEE International Image Sensor Workshop IIS07*, (Ogunquit, ME).

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 1297–1301. doi: 10.1021/nl904092h

Liu, S., van Schaik, A., Minch, B., and Delbruck, T. (2010). "Event-based 64-channel binaural silicon cochlea with q enhancement mechanisms," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, (Paris: IEEE).

Noack, M., Partzsch, J., Mayr, C., Henker, S., and Schuffny, R. (2010). "Biology-derived synaptic dynamics and optimized system architecture for neuromorphic hardware," in *Mixed Design of Integrated Circuits and Systems (MIXDES), 2010 Proceedings of the 17th International Conference*, (Warsaw).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH ARTICLE
published: 25 November 2014
doi: 10.3389/fnins.2014.00377

Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels

Saeed Afshar[1]*, Libin George[2], Jonathan Tapson[1], André van Schaik[1] and Tara J. Hamilton[1,2]

[1] Bioelectronics and Neurosciences, The MARCS Institute, University of Western Sydney, Penrith, NSW, Australia
[2] School of Electrical Engineering and Telecommunications, The University of New South Wales, Sydney, NSW, Australia

Edited by:
Chiara Bartolozzi, Italian Institute of Technology, Italy

Reviewed by:
Sadique Sheik, University of Zurich, ETH Zurich, Switzerland
Mattia Rigotti, IBM Research, USA
Stephen Nease, Cognitive Interaction Technology - Center of Excellence Bielefeld, Germany

****Correspondence:***
Saeed Afshar, Bioelectronics and Neuroscience, The MARCS Institute, University of Western Sydney, Locked Bag 1797, Penrith, NSW 2751, Australia
e-mail: s.afshar@uws.edu.au

This paper describes the Synapto-dendritic Kernel Adapting Neuron (SKAN), a simple spiking neuron model that performs statistical inference and unsupervised learning of spatiotemporal spike patterns. SKAN is the first proposed neuron model to investigate the effects of dynamic synapto-dendritic kernels and demonstrate their computational power even at the single neuron scale. The rule-set defining the neuron is simple: there are no complex mathematical operations such as normalization, exponentiation or even multiplication. The functionalities of SKAN emerge from the real-time interaction of simple additive and binary processes. Like a biological neuron, SKAN is robust to signal and parameter noise, and can utilize both in its operations. At the network scale neurons are locked in a race with each other with the fastest neuron to spike effectively "hiding" its learnt pattern from its neighbors. The robustness to noise, high speed, and simple building blocks not only make SKAN an interesting neuron model in computational neuroscience, but also make it ideal for implementation in digital and analog neuromorphic systems which is demonstrated through an implementation in a Field Programmable Gate Array (FPGA). Matlab, Python, and Verilog implementations of SKAN are available at: http://www.uws.edu.au/bioelectronics_neuroscience/bens/reproducible_research.

Keywords: spiking neural network, neuromorphic engineering, spike time dependent plasticity, stochastic computation, dendritic computation, unsupervised learning

INTRODUCTION

PRIOR WORK

Real neurons, the electrically excitable cells of the Eumetazoan, constitute an extremely diverse intractably complex community whose dynamic structures and functions defy all but the broadest generalizations (Herz et al., 2006; Llinas, 2008). In order to minimize this complexity, the field of Artificial Neural Networks (ANN) has traditionally modeled neurons as deterministic, centrally clocked elements which operate on real valued signals (Yegnanarayana, 1999). These signals represent neuronal rate coding where the spiking rate of a neuron encodes useful information and the adjustment of synaptic weights results in learning. This scheme, while mathematically amenable incurs a significant energy cost by discarding the rich temporal information available in the real signals used by neurons to communicate (Levy and Baxter, 1996; Laughlin, 2001; Van Rullen and Thorpe, 2001). In contrast, the highly optimized, low power, portable signal processing, and control system that is the brain readily uses temporal information embedded in the input signals and internal dynamics of its stochastic heterogeneous elements to process information (Xu et al., 2012).

More recently, the greater efficiency, higher performance, and biologically realistic dynamics of temporal coding neural networks has motivated the development of synaptic weight adaptation schemes that operate on temporally coding Spiking Neural Networks (SNN) (Jaeger, 2001; Maass et al., 2002; Izhikevich, 2006; Kasabov et al., 2013; Tapson et al., 2013; Gütig, 2014). After proposition many of these models are followed soon by their implementation in neuromorphic hardware (Mitra et al., 2009; Indiveri et al., 2011; Beyeler et al., 2013; O'Connor et al., 2013; Chicca et al., 2014; Rahimi Azghadi et al., 2014). One of the problems faced by neuromorphic hardware engineers is the hardware inefficiency of many neural network algorithms. These algorithms are almost always initially designed for performance in a constraint free mathematical context with numerous all-to-all connected neurons and/or to satisfy some biological realism criteria, which create difficulties in hardware implementation.

Additionally in order for such spiking systems to combine temporal coding and weight adaptation, multiple synapses consisting of synaptic transfer functions (or synaptic kernels) as well as synaptic weights must be realized for every input channel as shown in **Figure 1**. With the aim of being biologically plausible, exponentially decaying functions are typically chosen as the synaptic kernel, which is then multiplied by the synaptic weight. Such functions and weights are quite complex and difficult to implement in simple scalable analog and digital hardware with even the simplest schemes requiring at least one multiplication operation at every synapse. The difficulty of realizing multipliers at the synapse and the large number of synapses used in most algorithms has motivated moves toward more scalable digital synapses (Merolla et al., 2011; Seo et al., 2011; Arthur et al., 2012; Pfeil et al., 2012), novel memristor based solutions (Indiveri et al., 2013; Serrano-Gotarredona et al., 2013) and second order solutions such as sparse coding (Kim et al., 2014), time multiplexing,

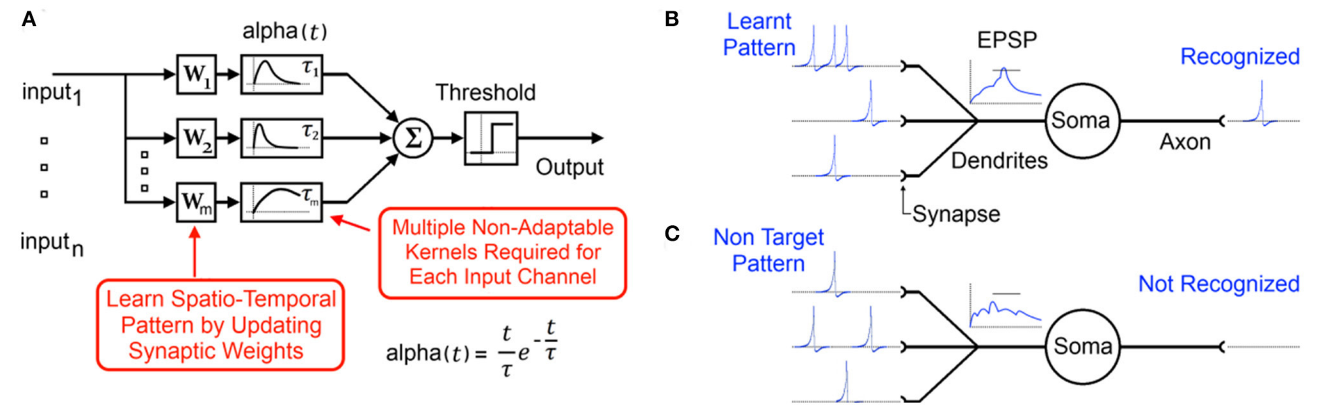


FIGURE 1 | (A) Typical functional model of a spiking neuron with static synaptic transfer functions that provide memory of recent spikes. **(B)** Biological representation of the neuron showing the learnt input spike pattern, the resultant Excitatory Post-Synaptic Potentiation (EPSP) and the output spike indicating pattern recognition. **(C)** Presentation of a non-target pattern results in an EPSP that does not cross the threshold producing no output spike.

and Address Event Representation (AER) (Zamarreno-Ramos et al., 2013) where only one or a few instances of the complex computational units are realized and these are utilized serially. Despite the success of these approaches such serial implementations can sometimes introduce associated bottlenecks, which can detract from the main strength of the neural network approach: its *distributed* nature (Misra and Saha, 2010).

Rather than implement complex synaptic weight adaptation, other neuromorphic SNN implementations have, in the last 3 years, focused exclusively on adjustment of explicit propagation delays along the neural signal path and coincidence detection of input spikes to encode memory (Scholze, 2011; Sheik et al., 2012, 2013; Dowrick et al., 2013; Hussain et al., 2014; Wang et al., 2014). This discarding of synaptic weights and kernels significantly simplifies implementation and improves scalability. The disadvantage is that explicit delay learning schemes can produce "sharp" systems with poor tolerance for the dynamically changing temporal variance they inevitably encounter in applications where neuronal systems are expected to excel: noisy, dynamic, and unpredictable environments.

One of the features shared by all the preceding systems is that the kernels used for encoding temporal information are static as shown in **Figure 2**. However recent advances in neurophysiology have revealed that synapto-dendritic structures and their associated transfer functions are highly complex and adapt during learning in response to the statistical contexts of their stimulus environment (Losonczy et al., 2008; Yoshihara et al., 2009; Kasai et al., 2010a; Lee et al., 2012; Rochefort and Konnerth, 2012; Smith et al., 2013; Colgan and Yasuda, 2014). These discoveries are significant in the context of the computational power of even single biological neurons. Whereas in the traditional neuron model synapto-dendritic structures function as weights and cables connecting one soma to the next, the recent findings have demonstrated a wide range of signal integration and processing occurring along the signal path, which confers considerable computational power to single neurons (Spruston, 2008; Silver, 2010; Harnett et al., 2012; Papoutsi et al., 2014). These effects represent novel dynamics with as yet unexplored emergent computational properties, which may potentially solve currently intractable problems in computational neuroscience (Bhatt et al., 2009; Shah et al., 2010). These dendritic adaptation effects have recently been modeled through large rule sets (Yu and Lee, 2003; Kasai et al., 2010b; Brunel et al., 2014) and in the neuromorphic field the use of dendrites for computation is beginning to be explored (Hsu et al., 2010; George et al., 2013; Ramakrishnan et al., 2013; Wang and Liu, 2013). However with biological realism as a major focus, many of the models carry significant extra complexity which can impede scalability.

NEURONS AS FUNCTIONAL MODELS OF DISTRIBUTED PROCESSING

In this paper the goal of performance in hardware motivates a change in focus from claims of accurate modeling of computation in biological neurons to exploiting the computational power of artificial but biologically inspired neurons. These are herein defined as a set of simple distributed informational processing units that communicate through binary valued pulses (spikes), receive inputs from multiple input channels (synapses and dendrites), and have a single output channel (axon).

Figure 3 illustrates the basic elements of SNN algorithms as well as some useful information flow and storage restrictions (red), which, if adhered to at the neuron design stage, prove helpful during the physical implementation stage. These restrictions include:

1. Self-contained: In a self-contained system, no external controlling system is required for the system to function. Examples of systems that are *not* self-contained include synapses that require adjustment via an external controller, or systems that assume an external supervisor in real world contexts where such a signal is unlikely to be available.
2. Scalable connectivity: Systems that require all-to-all connectivity between the neurons or where the synapses or dendrites directly communicate their weights or potentials to each other are not hardware scalable or biologically possible. All-to-all connected neurons require a geometrically increasing number of connections, which is prohibitive both in hardware and in the brain (Topol et al., 2006; Bullmore and Sporns, 2012).

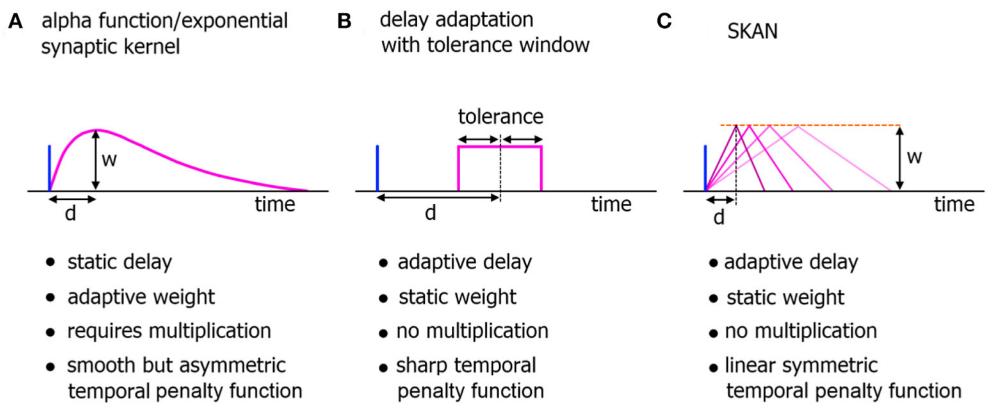

A alpha function/exponential synaptic kernel
w
d
time
• static delay
• adaptive weight
• requires multiplication
• smooth but asymmetric temporal penalty function
B delay adaptation with tolerance window
tolerance
d
time
• adaptive delay
• static weight
• no multiplication
• sharp temporal penalty function
C SKAN
w
d
time
• adaptive delay
• static weight
• no multiplication
• linear symmetric temporal penalty function

FIGURE 2 | Comparison of neuromorphic implementations of synapto-dendritic kernels. The characteristics of realized EPSP kernels are computationally important just prior to their being summed at the soma. These kernels represent the penalty function used to translate the temporal error in spatiotemporal spike patterns at the synapse to the membrane potential at the soma. Due to their large numbers, the complexity, functionality, and hardware cost of these kernels are a critical features of neuromorphic spiking networks. **(A)** The biologically plausible alpha function with adaptive weights. The delay of the kernel is static. **(B)** A neuromorphic delay learning system with a temporal tolerance window. **(C)** The adaptable kernel of SKAN with adaptive delay when the kernel peak value/synaptic weight w is kept constant as is the case in this work.

3. Storage of time series data: Systems whose processing units require large segments of their time series data to be stored and be accessible for later processing in the fashion of standard processors require a significant amount of on-site memory not possible in biological systems and would add significant complexity to neuromorphic hardware. Furthermore, such systems overlap the domain of distributed processors such as GPUs and fall outside the neuromorphic scope.
4. Multiplication: Multipliers are typically inefficient to implement in hardware and are limited in standard digital solutions such as Field Programmable Gate Arrays (FPGAs) and Digital Signal Processors (DSPs). Their computational inefficiency and their limited number available on a hardware platform result in neural networks implemented with time-multiplexing. This, in turn, limits the size and the applications where this hardware is viable (Zhu and Sutton, 2003; Pfeil et al., 2012).

MATERIALS AND METHODS

The elements of Synapto-dendritic Kernel Adapting Neuron (SKAN) and its learning rule are defined in the first part of this section. In the second part, the dynamical behaviors of SKAN are described.

SKAN BUILDING BLOCKS

At the single neuron level, SKAN consists of a combined synapto-dendritic kernel adaptation and a homeostatic soma with an adapting threshold as shown in **Figure 4**.

Synapse/dendrite

An incoming input spike initiates a simplified synapto-dendritic kernel at each input channel i. This kernel is controlled by a physiological process, p_i, and for simplicity is modeled as a ramp up and a ramp down sequence generated via an accumulator r_i with step size Δr_i. An input spike triggers p_i, starting the first phase where the accumulator ramps up at each time step Δt by Δr_i until it reaches a maximum value w_i which represents the synaptic weight, and which is kept constant throughout this paper to simplify the algorithm. After r_i reaches w_i, the process switches from the ramp up phase, $p_i = 1$, to a ramp down phase, $p_i = -1$, which causes the accumulator to count down at each time step toward zero with the same step size Δr_i, until it reaches zero, turning off the physiological process, $p_i = 0$. It will stay in this state until a new incoming spike re-initiates the sequence. This simple conceptual sequence, which is analogous to a dendritically filtered neuronal EPSP, is illustrated in **Figure 5**.

The state of the ramp up ramp down flag sequence is described by Equation 1:

$$p_i(t) = \begin{cases} 1 & \text{if } (u_i(t) = 1 \wedge p_i(t-1) = 0) \\ & \vee (p_i(t-1) = 1 \wedge r_i(t-1) < w_i) \\ -1 & \text{if} (p_i(t-1) = 1 \wedge r_i(t-1) \geq w_i) \\ & \vee (p_i(t-1) = -1 \wedge r_i(t-1) > 0) \\ 0 & \text{else} \end{cases} \quad (1)$$

The w parameter in SKAN has similarities to the weight by which a synaptic kernel is multiplied in standard synaptic STDP neuron models and neuromorphic circuits, but with the advantage of not requiring any multipliers, which are otherwise required at every synapse in hardware implementations. The adjustment of w in SKAN, via standard synaptic STDP schemes would allow synaptic prioritization and/or the closing off of inactive or noisy channels. The combined effects of dendritic structure and synaptic weight plasticity has only recently begun to be explored, but early evidence points to significant computational power of such a combined system (Sjöström et al., 2008). In this paper, however, in order to clearly demonstrate the stand-alone capabilities of SKAN's synapto-dendritic kernel adaptation mechanism, the synaptic weight parameter of w is held constant and is identical for all synapses.

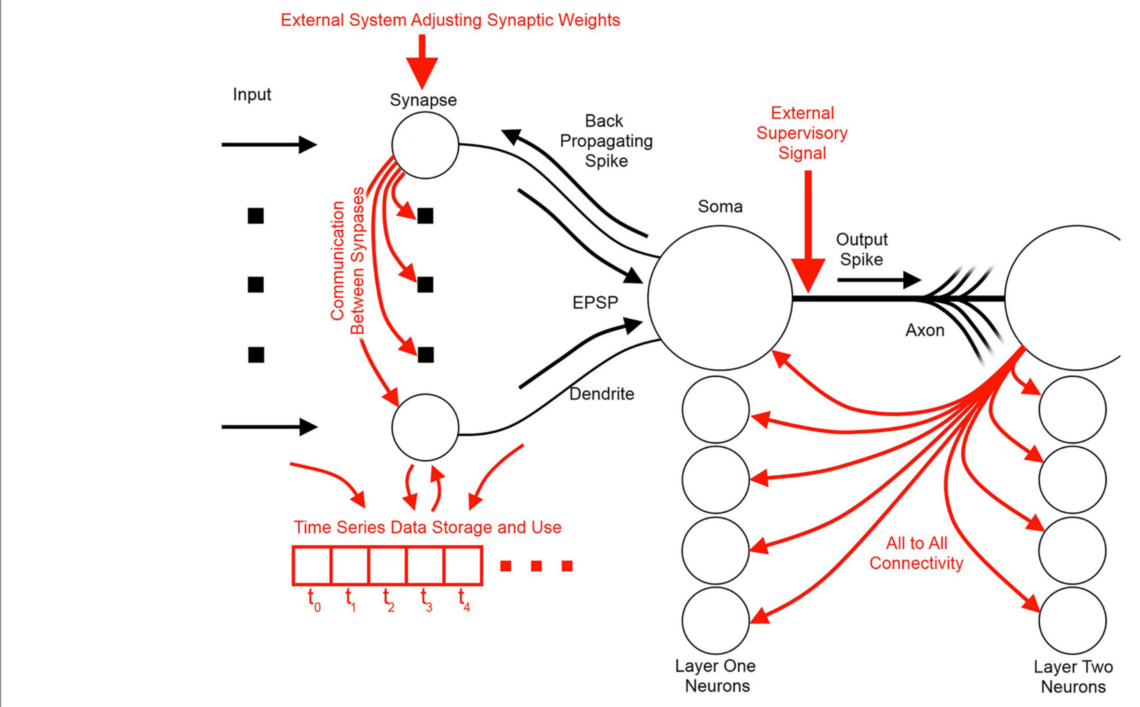


FIGURE 3 | Information flow schemes in unsupervised spiking neural network algorithms and their impact on hardware implementation. Black indicates the fundamental elements and information paths of a spiking neural network. Red indicates added features and information paths that can cause difficulties in hardware, or limit algorithm utility.

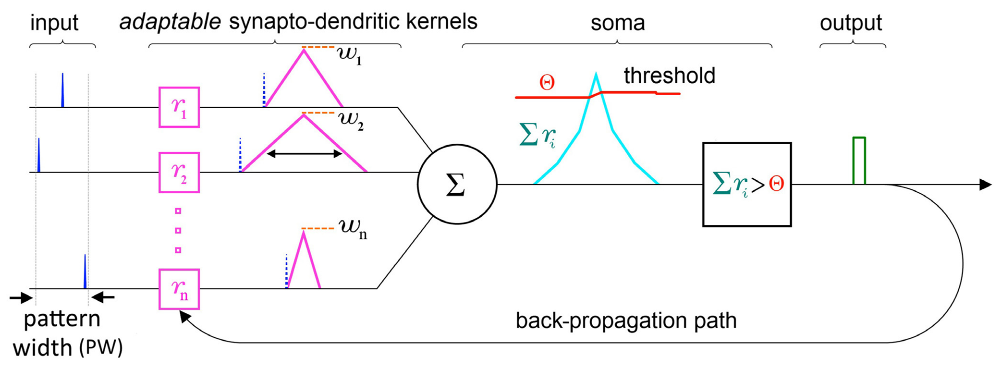


FIGURE 4 | Schematic of the elements and information paths in a SKAN neuron. The input spikes (blue) trigger adaptable synapto-dendritic kernels (magenta) which are summed to form the neuron's somatic membrane potential (cyan). This is then compared to an adaptive somatic threshold (red) which, if exceeded, results in an output pulse (green). The output pulse also feeds back to adapt the kernels. Note that in this paper the synaptic weights (orange) are held constant and equal for all synapses. Also note that the back propagating signal does not travel beyond the synapto-dendritic structures of the neuron to previous neural layers.

Soma

At the soma the synapto-dendritic kernels are summed together. This summed term is analogous to the membrane potential of a biological neuron. Along with the membrane potential the soma uses a dynamic threshold voltage parameter $\Theta(t)$ and as long as the membrane potential exceeds threshold, the soma spikes, setting the binary $s(t)$ from 0 to 1 as described in Equation 2:

$$s(t) = \begin{cases} 1 & \text{if } \sum_i r_i(t) > \Theta(t-1) \\ 0 & \text{else} \end{cases} \quad (2)$$

SKAN differs from most previous spiking neuron models in not resetting the membrane potential after spiking (see Denève, 2008; Tapson et al., 2013 for exceptions). This permits wide pulse widths at the neuron output $s(t)$. While such wide pulses do

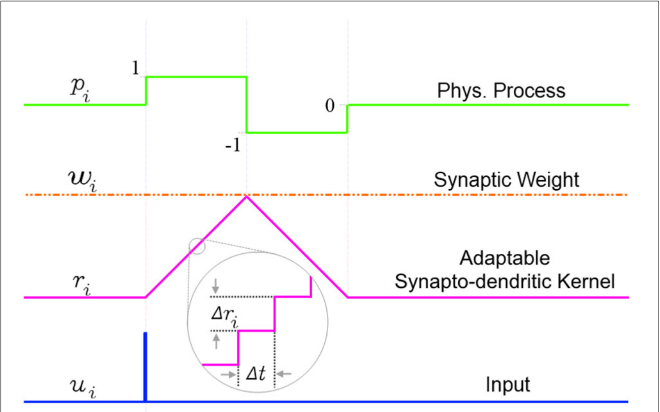


FIGURE 5 | The simplified adaptable synapto-dendritic kernel of SKAN. An input spike (blue) triggers the kernel's ramp up ramp down sequence. The input spike sets a flag p_i representing a physical process to one (green). The flag causes an accumulator (magenta) to count up from zero by Δr_i at each time step until it reaches w_i (constant orange dotted line), after which the flag is set to negative one, which causes the accumulator to count back down to zero, at which point the flag returns to zero completing the sequence. The value of the accumulator represents the synapto-dendritic kernel, i.e., the post-synaptic potential, which travels to the soma and is summed with other kernels to produce the somatic membrane potential.

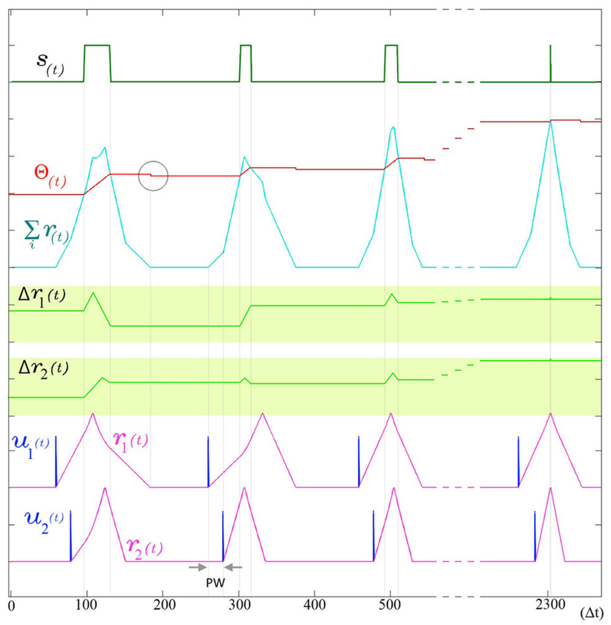


FIGURE 6 | The adaptation of SKAN. The kernels and the threshold of SKAN adapt in response to repeated spatio-temporal pattern presentations. For visual clarity the pattern only consists of the Inter-Spike Interval (ISI) across two input channels $u_i(t)$ such that the pattern width (*PW*) is equivalent to the ISI. By the third presentation of the pattern the kernels have captured the ISI information. With each subsequent presentation the threshold $\Theta(t)$ increases making the neuron more selective as the kernel step sizes $\Delta r_i(t)$ increase making the kernels narrower. As a result each pattern presentation increases the neuron's confidence about the underlying process producing the ISI's, narrowing the neuron's receptive field around the target ISI and producing a smaller output pulse $s(t)$ until, by the 11th presentation ($t = 2300\ \Delta t$), the Θ_{rise} during the output spike and Θ_{fall} balance each other such that the $\Theta_{before} \approx \Theta_{after}$. The soma output spike $s(t)$ is now a finely tuned unit delta pulse which indicates high certainty. When the membrane potential returns to zero, the neuron's threshold falls as indicated by the gray circle.

not resemble the canonical form of the single spike, they are analogous to concentrated spike bursts and play a significant part in the functioning of SKAN.

FEEDBACK MECHANISMS/LEARNING RULES

Synapto-dendritic kernel slope adaption

One of the central elements of SKAN is the feedback effect of the output pulse $s(t)$ on each of the synapto-dendritic kernels. Here $s(t)$ is analogous to the back propagating spike signal in biological neurons which travels back up the dendrites toward the synapses and is responsible for synaptic STDP.

The logic of the kernel adaptation rule is simple; if a particular dendrite is in the ramp up phase $p_i = 1$ and the back propagation signal $s(t)$ is active, the soma has spiked and this particular kernel is late to reach its peak, meaning that the other kernels have cooperatively forced the membrane potential above the threshold while this kernel has yet to reach its maximum value w_i. In response, the ramp's step size Δr_i is increased by some small positive value ddr for as long as the output pulse is high $[s(t) = 1]$ and the kernel is in the ramp up phase. Similarly if a kernel is in the ramp down phase $p_i = -1$ when the back propagation signal is high, then the kernel peaked too early, having reached w_i and ramping down before the neuron's other kernels. In this case the ramp step size Δr_i is decreased by ddr. Equation 3 describes this simple kernel adaptation rule:

$$\begin{bmatrix} r_i(t) \\ \Delta r_i(t) \end{bmatrix} = \begin{bmatrix} r_i(t-1) \\ \Delta r_i(t-1) \end{bmatrix} + p_i(t-1) \begin{bmatrix} \Delta r_i(t-1) \\ ddr \times s(t-1) \end{bmatrix} \quad (3)$$

The use of indirect evidence about the dynamic state of other dendrites in the form of the back propagating spike is a central feature in the operation of SKAN and enables the synchronization of all the neuron's dendritic kernel peaks as shown in **Figure 6**.

Threshold adaptation

The threshold of SKAN is adaptive and changes under two conditions: when the neuron outputs a spike and when the membrane potential returns to zero.

At every time step during an output pulse $s(t) = 1$ the threshold increases by Θ_{rise}. This increase in the threshold is analogous to the frequency adaptation effect seen in neurons, which creates a feedback loop reducing the ability of the neuron to spike. Similarly in SKAN, the higher threshold reduces the likelihood and duration of an output pulse. This effect is shown in **Figure 6** and described in the first line of Equation 4.

$$\Theta(t) = \begin{cases} \Theta(t-1) + \Theta_{rise} & \text{if } \sum_i r_i(t) > \Theta(t-1) \\ \Theta(t-1) - \Theta_{fall} & \text{if } \sum_i r_i(t) = 0 \wedge \sum_i r_i(t-1) > 0 \\ \Theta(t-1) & \text{else} \end{cases} \quad (4)$$

The post spike decrease in threshold Θ_{fall} operates in opposition to the Θ_{rise} term. The returning of the membrane potential $\Sigma r_i(t)$ to zero causes a decrease in the threshold by Θ_{fall} as described by the second line of Equation 4 and shown in **Figure 6**. The counter balancing effect produced by the Θ_{fall} and Θ_{rise} in SKAN is a highly simplified version of the complex mechanisms underlying spike-threshold and frequency adaption in biological neurons (Fontaine et al., 2014; Lee et al., 2014), where excited neurons eventually reach an equilibrium state through homeostatic processes such that the average spike frequency of neurons with a constant input tends asymptotically toward a non-zero value as $t \to \infty$. This simple rule set describes all the elements of a single SKAN.

SINGLE SKAN DYNAMICS

In this section the dynamics emerging from SKAN's rule are discussed for the single neuron case.

Observing the first spike in a spike train or burst

As described in the first line of Equation 1 the ramp up phase of the kernel at channel i is only initiated if a spike arrives at the channel ($u_i = 1$) while and the kernel is inactive ($p_i = 0$). As a result while the ith kernel is active no further input spikes are observed. This has the effect that for each input channel the neuron trains on the first spike of a spike train or burst. For the case where the spike train or burst is of shorter duration than the total duration of the kernel, the behavior of the neuron is identical one where the burst is replaced by a single input spike arriving at the start of the burst. The effect of more general Poisson noise spikes is described later in this section.

Selecting to learn the commonest spatio-temporal patterns

As a single neuron, SKAN has previously been shown to select and learn the most common spatio-pattern presented in a random sequence containing multiple patterns (Sofatzis et al., 2014a). This effect has been demonstrated in the context of visual processing where hand gestures were transformed to spatio-temporal patterns via a neuronal transform operation (Afshar et al., 2013) and processed by SKAN (Sofatzis et al., 2014b). **Figure 7** shows the performance of a four input neuron as a function of spatio-temporal pattern probability. The graph shows that the neuron's selection of commonest pattern is significantly above chance such that for sequences with $P(\mathrm{x}) > 0.85$ only the more common pattern will selected.

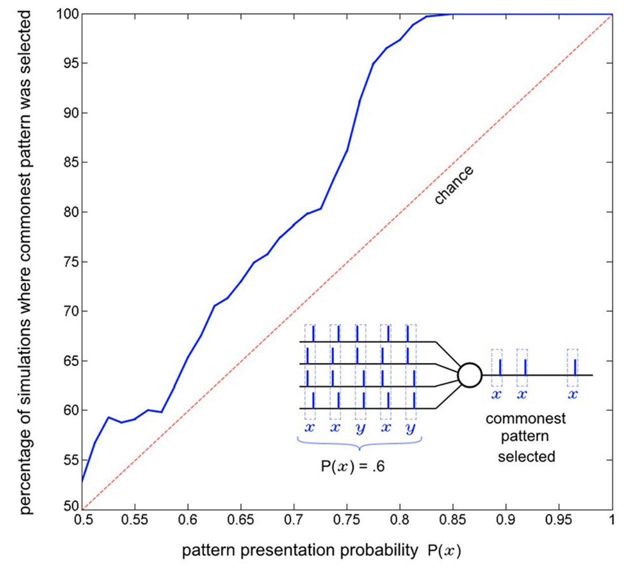


FIGURE 7 | Commonest pattern selection as a function of pattern presentation probability. The inset illustrates one simulation a 5 pattern long sequence where each pattern is sampled from two randomly initialized spatio-temporal patterns *x* and *y*, with probability $P(x) = 0.6$. In this particular simulation pattern *x* was selected by the neuron. The plot shows data resulting from the same experiment but with 1000 simulations of 300 pattern long sequences for each probability $P(x) = 0.5$ to 1. The graph shows that the likelihood of a pattern being selected rises with increasing presentation probability. For each simulation the output of the neuron for the second half of the sequence (150–300th pattern) was recorded and it was determined whether pattern *x* or pattern *y* had been selected. Also tested was whether both, or neither pattern was selected by the neuron at some point during the sequence (i.e., the neuron spiked at least once for both of the patterns or failed to spike for a pattern it had selected during the sequence). In the more than seven million pattern presentations ($1000 \times 150 \times 51$) neither of these occurred.

SKAN response time improves with adaptation without information loss

In addition to the kernel adaptation and increasing threshold effect, the response time of SKAN, i.e., the time from the last arriving input spike in a pattern to the neuron's output spike, decreases with every pattern presentation. This effect, shown in **Figure 8**, is absent in the standard STDP schemes where improved response times comes at the cost of information loss. In STDP schemes the earliest spikes in a spatio-temporal pattern tend to be highly weighted while the later spike lose weight and have little effect on recognition (Masquelier et al., 2009). This behavior can be seen as advantageous if an assumption is made that the later spikes in carry less information however in this is an assumption that cannot be made in general. In contrast SKAN's adaptable kernels reduce output spike latency with adaptation while still enabling every spike to affect the output. This effect proves critical in the context of a multi-SKAN competitive network, where the best-adapted neuron is also always the fastest neuron to spike.

As shown in **Figure 8**, the combination of the kernel and threshold adaptation rules of SKAN increases Δr and decreases the response time between the last arriving input spike and the rising edge of the output spike with each presentation. If this increase is left unchecked Δr will increase until it equals w at which point the kernels take the shape of a single pulse such that $T_\infty = 1\Delta t$. To prevent this Δr must saturate at Δr_{max} as shown in **Figure 8** with Δr_{max} limited by Equation 5. This restriction ensures that the kernel of the first spike in an input pattern cannot return to zero before the last spike in the pattern arrives enabling all kernels to converge due to feedback from the same output signal.

$$\Delta r_{max} < w/PW \tag{5}$$

where PW is the maximal pattern width of the target pattern.

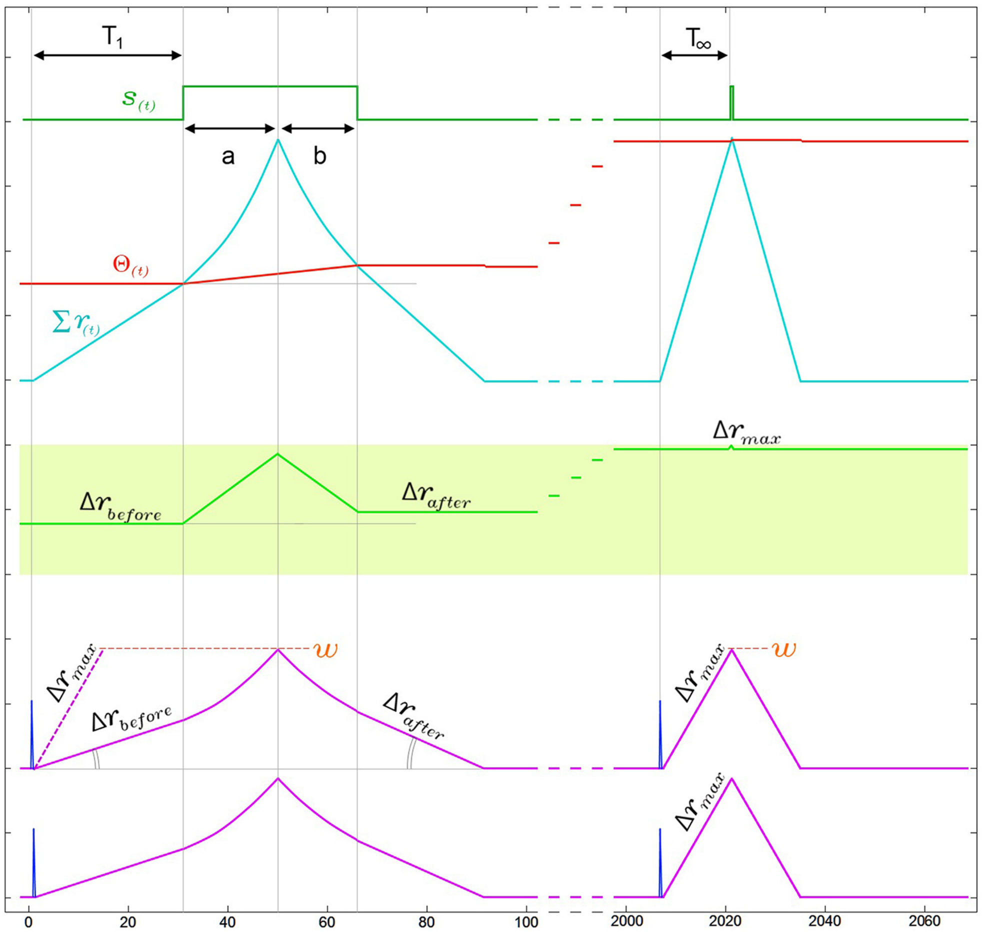


FIGURE 8 | Narrowing of kernels leads to improved response time during neuronal adaptation in a two input neuron. For visual clarity the neuron is presented with an ISI $= 0\Delta t$ pattern and the two kernels start with identical initial slopes $[\Delta r_1(0) = \Delta r_2(0) = \Delta r_{before}]$. In the region under the output pulse, $r(t)$ is the second integral of the constant *ddr* and therefore follows time symmetric parabolic paths (a) and (b) as it rises and falls. However due to the threshold rise which also occurs during the output pulse, the output pulse is not symmetric around the $r(t)$ peak, such that the parabolic ramp down phase (b) is shorter than the parabolic ramp up phase (a). As a result of this asymmetry Δr_{after} is larger than Δr_{before}. This effect increases the kernel's slope Δr with each pattern presentation, narrowing the kernels until Δr reaches Δr_{max}. As a result of this narrowing, the response time of the neuron from last arriving input spike to the rising edge of the output, which is T_1 in the first presentation, improves until it reaches its minimal possible value $T_\infty \approx w/\Delta r_{max}$.

Evolution of the temporal receptive field in SKAN approximates statistical inference

Recent work has demonstrated the connection between synaptic weight adaptation and approximate probabilistic inference in the context of rate coding and spiking networks (Bastos et al., 2012; Boerlin et al., 2013; Pouget et al., 2013; Corneil et al., 2014; Kappel et al., 2014; Kuhlmann et al., 2014; Paulin and van Schaik, 2014; Tully et al., 2014), where typically the state of binary hidden variables are inferred from noisy observations using a large number of neurons. In this section we show that synapto-dendritic kernel adaptation enables a single neuron to make statistical inferences not about binary hidden variables but about hidden ISI generating processes. **Figure 9** illustrates the evolution of the temporal receptive field of a neuron with two inputs as the neuron attempts to learn the statistics of an underlying process that produces ISIs with linearly increasingly temporal jitter. The receptive field of the neuron describes the amount by which the membrane potential $\Sigma r_i(t)$ exceeds the threshold $\Theta(t)$ as a function of the input spike pattern times of $u_i(t)$. For the simple two input case illustrated, the receptive field is a scalar function of the one-dimensional ISI. In order to calculate the receptive field, following each pattern presentation the neuron's new parameters (Δr_i and Θ) were saved and the neuron was simulated repeatedly using these saved parameters for every possible ISI given the maximum pattern width *PW*. For each simulation the summation in Equation 6 was calculated at the end of the simulation resulting in the receptive fields shown in **Figure 9**.

$$RF_{i=2}(\tau) = \sum_t \left(\sum_i r_i(\tau, t) - \Theta(\tau, t)\right) \times s(\tau, t) \qquad (6)$$

where τ is the ISI being simulated.

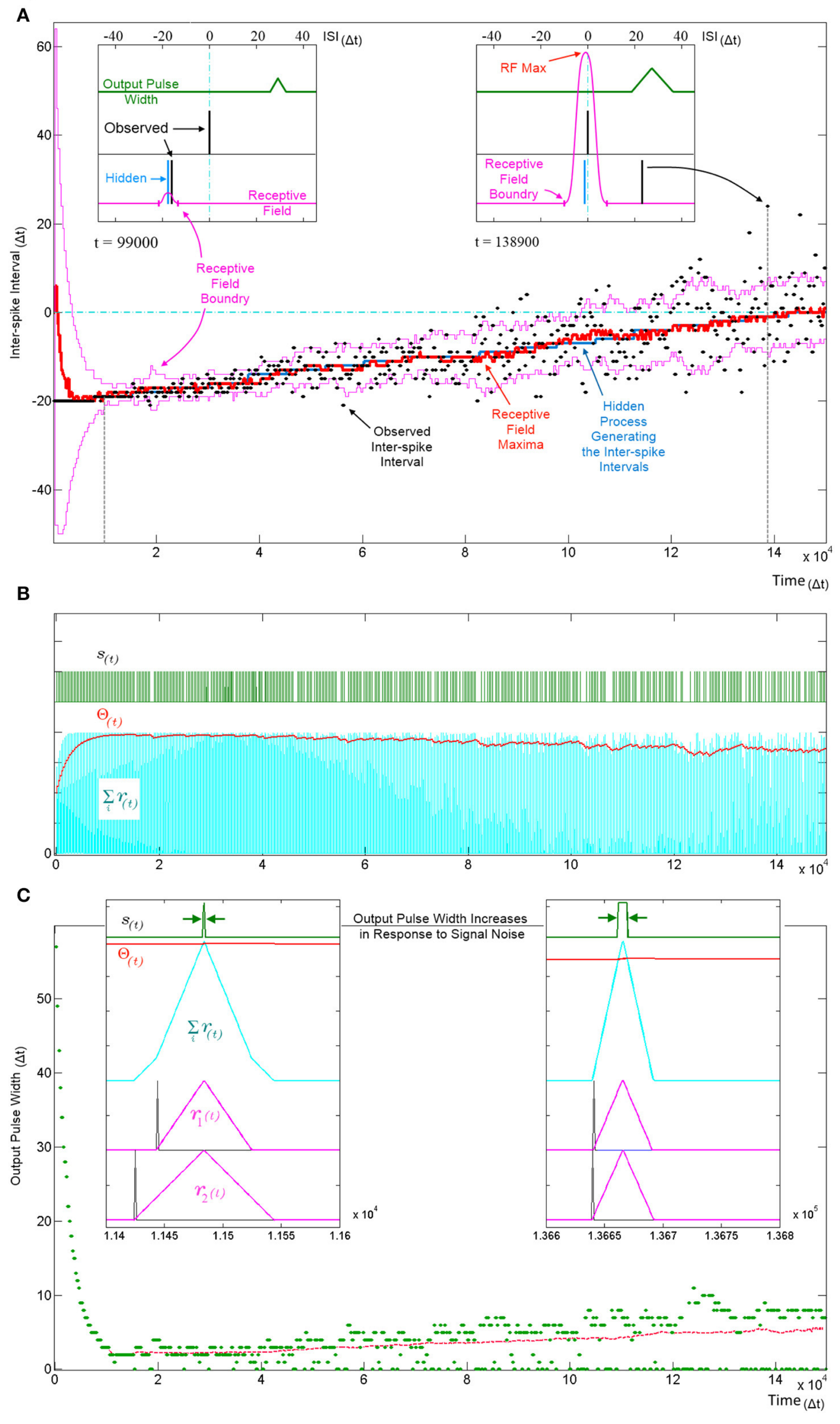


FIGURE 9 | Tracking a hidden ISI producing process and its variance. All three panels **(A–C)** show different aspects of the same simulation where a single SKAN learns statistics of a dynamic ISI across two input channels. **(A)** A hidden process (blue) moves from ISI = $-20\ \Delta t$ to ISI = $0\ \Delta t$. The process begins with no temporal jitter noise, such that the observed ISI's (black dots) equal the hidden process ($\sigma = 0\Delta t$) and the blue hidden process is covered by the observed black dots. At $t = 0$, the sum of the neuron's randomly

(Continued)

FIGURE 9 | Continued
initialized kernels peak at ISI $= 6\ \Delta t$. As the kernels adapt and capture the ISI information, the receptive field maximum (red line) converges on to the observed ISIs. This causes the threshold to rise shrinking the receptive field to a minimum size (left inset $t = 9.9\ \text{k}\Delta t$). At this stage receptive field boundaries (pink lines) lie very close on either side of the hidden process' mean value. As the simulation continues the noise in the ISI producing process is deliberately increased linearly with time. The neuron continues to follow the process but with every spike that doesn't land on the receptive field maximum the threshold falls slightly which increases the receptive field size and increases the neuron's receptivity to ever more unlikely observations (right inset $t = 138.9\ \text{k}\Delta t$). **(B)** Variables and behavior of the neuron throughout the simulation: after a rapid initial increase the threshold $\Theta(t)$ settles near the peak of the membrane potential $\Sigma r_i(t)$. As the noise increases linearly the threshold begins to fall gradually. Missing output spikes in the $s(t)$ spike train correspond to input spikes that have landed outside the receptive field boundaries. **(C)** The output pulse width (green = observed, red = running average) increases with increasing signal noise. As with the receptive field size, the output pulse width is initially large. As more patterns are observed, the threshold rises and settles just below the peak of the membrane potential and the pulse width reaches a minimum width of 1–2 Δt. At this low noise level ($\sigma \approx 0.5\Delta t$) there are no missing outputs, such that all pulse widths are above zero. As the noise increases, more ISIs land away from the receptive field maximum and some fall completely outside the receptive field, decreasing the threshold, which results in wider output pulse width whenever observed ISI's *do* land near the receptive field maximum. The dashed magenta line tracks the mean spike width, which also increases with noise. This illustrates that the mean output pulse width of SKAN is a reliable correlate of input noise level.

The ISI at which the receptive field expression above is at its maximum (*RF* Max) indicates the ISI for which the neuron is most receptive and may be interpreted as the ISI expected by the neuron. Similarly the ISI boundary where the receptive field expression goes to zero is the limit to the range of ISI's expected by the neuron. An ISI falling outside the receptive field boundaries results in no spike and no adaptation but simply reduces the neuron's confidence and can be viewed as outlier.

Figure 9A shows SKAN's receptive fields tracking the statistics of a moving ISI generating process with dynamic noise levels with a high level of accuracy such that the blue line indicating the hidden process is barely visible from under the red line marking the receptive field maximum. **Figure 9C** shows the neuron transmitting wider output or bursts with increasing noise. In addition, increasing ISI noise causes a growing gap between the envelope of the pulse widths and the running average of the pulse widths. This increasing gap is critical to the operation of the neuron, as it is caused by missed pattern presentations, i.e., patterns that produce no output pulse because of the presented noisy pattern being too dissimilar to the one the neuron has learnt and expects. The effect of a missed pattern is a fall in the neuron's threshold by Θ_{fall}. When presented with noiseless patterns this fall would be balanced almost exactly by the threshold rise due to the Θ_{rise} term in Equation 4 during the output pulse. However, without the output spike there is a net drop in threshold. Yet this lower threshold also makes the neuron more receptive to noisier patterns creating a feedback system with two opposing tendencies which:

1. Progressively narrows kernels around the observed input pattern while shrinking the neuron's receptive field by raising the threshold.
2. Expands the receptive field in response to missed patterns by reducing the threshold while allowing the kernels to learn by incorporating ever less likely patterns.

The balance between these two opposing tendencies is determined by the ratio Θ_{rise}:Θ_{fall}, which controls how responsive the neuron is to changing statistics. With a stable noise level SKAN's dynamics always move toward an equilibrium state where the neuron's tendency to contract its receptive field is precisely balanced by the number of noisy patterns not falling at the receptive field maximum. This heuristic strategy results in the receptive field's maximum and extent tracking the expected value of the input ISI's and their variance respectively as shown in **Figure 10**.

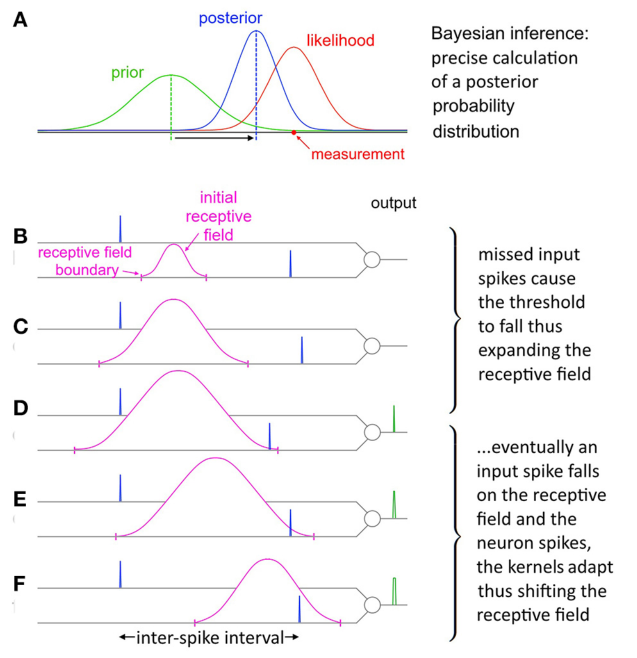

A
posterior
likelihood
prior
measurement
Bayesian inference: precise calculation of a posterior probability distribution
B
initial receptive field
receptive field boundary
output
C
D
E
F
missed input spikes cause the threshold to fall thus expanding the receptive field
...eventually an input spike falls on the receptive field and the neuron spikes, the kernels adapt thus shifting the receptive field
←inter-spike interval→

FIGURE 10 | Evolution of SKAN's receptive field in response to input. **(A)** Total resultant change in SKAN's receptive field after multiple pattern presentations. **(B)** SKAN with a small initial receptive field which does not match the ISI distribution. The input spike lands outside the receptive field boundaries. **(C)** As more ISI's fall outside the small receptive field the threshold falls and the receptive field expands, but without shifting the position of its maximum value. **(D)** An ISI just falls on to the greatly expanded receptive field producing an output spike. **(E)** The output spike causes the SKAN kernels to adapt shifting the receptive field toward the true position of the underlying process. **(F)** As more and more ISI's fall closer to the receptive field maximum wider output pulses are produced which adapt the kernels faster shifting the receptive field more rapidly while the resultant rise in the threshold contracts the receptive field. With enough observations the receptive field would eventually become centered on the input ISI distribution with the receptive field boundaries tracking the ISI's distribution.

LEARNING IN THE PRESENCE OF POISSON SPIKE NOISE AND MISSING TARGET SPIKES

In addition to robustness to temporal jitter in the put pattern an important feature of neural systems is their performance in the presence of Poisson spike noise. Recent work has highlighted that unlike most engineered systems where noise is assumed to degrade performance, biological neural networks can often utilize such noise as a resource (McDonnell and Ward, 2011; Hunsberger et al., 2014; Maass, 2014). In the neuromorphic context the performance of neural network architectures in the presence of noise is well documented (Hamilton and Tapson, 2011; Hamilton et al., 2014; Marr and Hasler, 2014). To test SKAN's potential performance in stochastic real world environments, the combined effects of extra noise spikes as well as missing target spikes needs to be tested. **Figure 11** illustrates how different signal to noise ratios can affect SKAN's ability to learn an embedded spatio-temporal spike pattern.

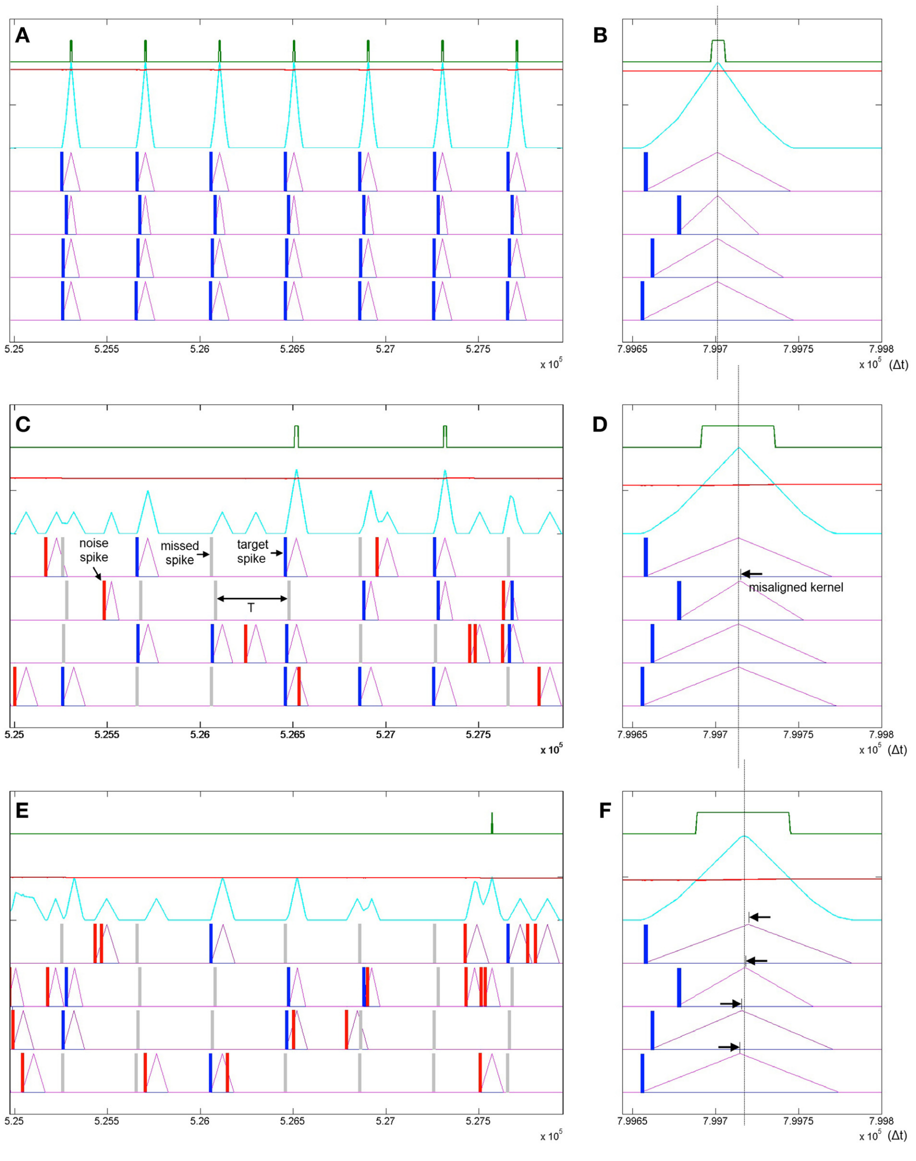


FIGURE 11 | Learning spatio-temporal spike patterns in the presence of both Poisson spike noise and missing target spikes. Panel **(A)** shows the presentation of seven patterns in the middle of a simulation sequence with a noiseless environment. The kernels are highly adapted ($\Delta r_2 = \Delta r_{max}$), the threshold is high and the output spikes are narrow indicating high certainty. Panel **(B)** shows the result of a final noiseless test pattern at the end of the simulation showing in detail that the kernels resulting from the test pattern peak at the same time. Panel **(C)** shows the same interval of the same simulation as panel **(A)** but with a 1:1 signal to noise ratio where the probability of a target spike being deleted is half or P(signal) = 0.5 and the Poisson rate is also half such that P(noise) = 0.5/T. Panel **(D)** shows the result of a noiseless test pattern presentation at the end of the simulation. The increased level of noise has resulted in an incorrect ramp step (Δr_2) such that the r_2 kernel peaks slightly late (black arrow). Panel **(E)** shows a simulation with a 1:2 signal to noise ratio. Panel **(F)** shows that the high noise level has resulted in slight misalignment of all four kernels.

To quantify the performance of SKAN in the presence of Poisson noise and missing target spikes a series of simulations each comprising of 2000 pattern presentations were performed. At the end of each simulation the RMS error between the neuron's receptive field maxima and the random target pattern was measured and is shown in **Figure 12**.

MULTI-SKAN CLASSIFIER

In order to extend a single learning neuron to a classifier network it is important that different neurons learn different patterns. Ideally a neuron in a layer should not be in anyway affected by the presentation of a pattern that another neuron in the same layer has already learnt or is better placed to learn.

As outlined in Equation 3, SKAN adapts its kernels only during an output pulse. This rule is particularly conducive to competitive learning such that the simple disabling of the neuron's spiking ability disables all learning. Whereas previously proposed algorithms utilize multi neuron Winner-Take-All layers with real valued rate based inhibitory signals to prevent correlated spiking and maximize the network learning capacity (Gupta and Long, 2009; Nessler et al., 2013), in a SKAN network a simple global inhibitory OR gate serves the same function. The reason a simple binary signal can be used here is that in a SKAN network the best-placed neuron for any pattern will be the fastest neuron to spike. This allows a layer of neurons with shared inputs to learn to recognize mutually exclusive spatio-temporal patterns. To this end, Equation 2, describing the neuron's output, is replaced by Equation 7 (underlined terms added). The addition of a global decaying inhibitory signal as described in Equation 8, act on all neurons to disable any rising edge at the output. This means that neuron n can only *initiate* an output spike s_n if no other neuron has recently spiked, i.e., the inhibitory signal is inactive $[inh(t-1)=0]$ and it can only *continue* spiking if it was already spiking in the last time step $[s_n(t-1)=0]$.

$$s_n(t) = \begin{cases} 1 & \text{if} \sum_i r_{n,i}(t) > \Theta_n(t-1) \\ & \underline{\wedge\ (inh(t-1)=0 \vee s_n(t-1)=1)} \\ 0 & \text{else} \end{cases} \tag{7}$$

$$inh(t) = \begin{cases} inh_{max} & \text{if} \bigcup_n s_n(t) = 1 \\ inh(t-1) - inh_{decay} & \text{if } inh(t-1) > 0 \\ 0 & \text{else} \end{cases} \tag{8}$$

As shown in **Figure 13** and described in Equation 8, the inhibitory signal is realized via an OR operation on the output of all neurons, and a decaying behavior which keeps the inhibitory signal active for a period of time after a neuron has spiked to prevent spiking by other neurons. After the output spike ends, this feedback loop decays from inh_{max} by inh_{decay} at each time step until reaching zero at which point the global inhibitory signal turns off allowing any neuron to spike. As shown in **Figure 13** the decay only begins at the end of the pulse making the inhibitory signal operate as

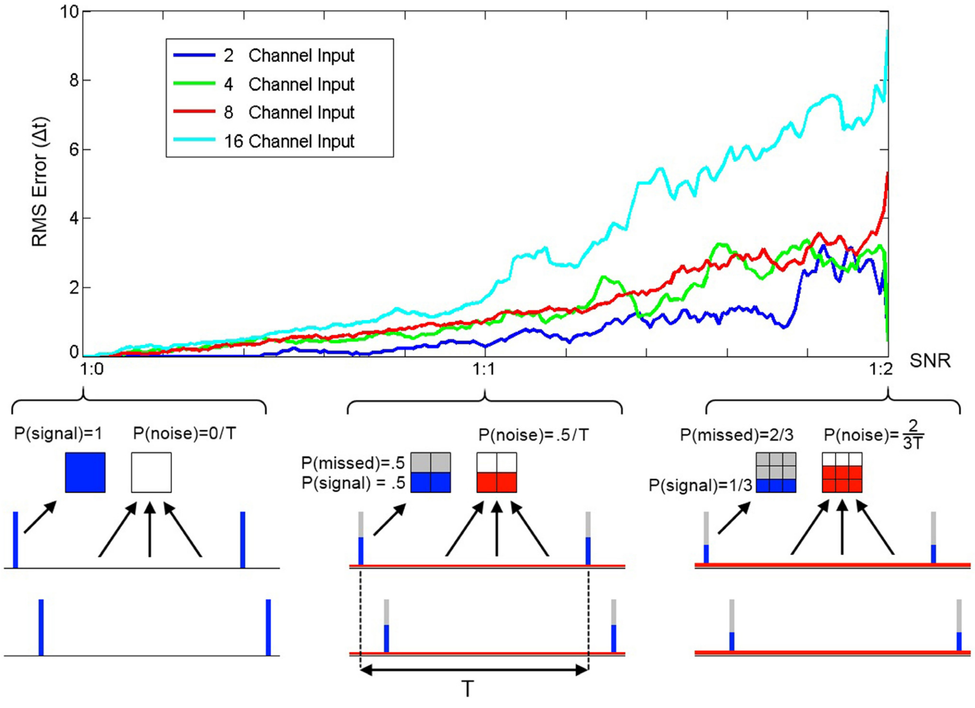


FIGURE 12 | RMS error between receptive field maxima and target spike patterns as a function of spike signal to noise ratio. The three bottom panels show the spike probability distributions at three points along the SNR axis. The signal spikes (blue), missed spikes (gray), and noise spikes (red) are illustrated for the three cases of 1:0, 1:1, and 1:2 signal to noise ratios. The mean spike rate was maintained at 1 spike per channel per time period between pattern presentations T. At the completion of a simulation with one thousand pattern presentations the RMS error between the resulting receptive field maxima and the target spatio-temporal pattern was calculated. As the plot illustrates the error increases with noise and simulations of neurons with more input channels resulted in higher error.

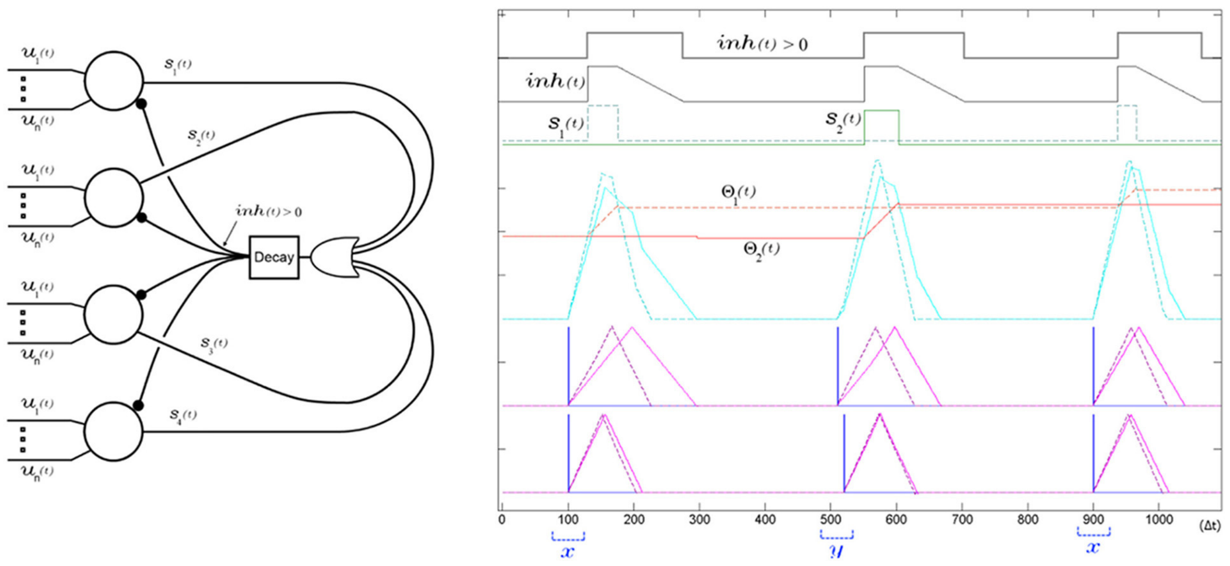


FIGURE 13 | A single global decaying inhibitory signal suffices to push apart the neurons' receptive fields and decorrelate the spiking of the SKAN network. Left panel shows the network diagram of four neurons with an inhibitory signal. The decay feedback loop extends the duration of the inhibitory signal beyond the initial triggering spike via the *inh*(*t*) signal using a counter and a comparator in the decay block. The right panel shows the simulation results from a two input two neuron network learning to classify two ISI's *x* and *y*. The sum of the randomly initialized kernels of neuron one (dashed) happen to peak earlier than neuron two (solid) so that neuron one fires first in response to the first pattern (*x* with ISI = 0 Δ*t*). During this first output pulse neuron one's threshold rises sharply reducing its receptivity, while its kernel step sizes adapt toward each other such that $\Delta r_{1,1} \approx \Delta r_{1,2}$. Meanwhile the inhibitory signal blocks neuron two from spiking when its kernel sum exceeds its threshold only a few time steps after neuron one, which means the neuron is prevented from adapting to pattern *x*. At the second pattern presentation pattern *y* is shown (ISI = 10 Δ*t*). For this pattern the sum of the kernels of the second neuron, still unchanged from their random initialization, reach that neuron's threshold slightly earlier than neuron one and so neuron two spikes and begins adapting to pattern *y*. A subsequent presentation of pattern *x* again triggers neuron one and the kernels of the two neurons increasingly fine tune to their respectively chosen pattern with each presentation as their thresholds rises reducing their receptivity to other patterns.

a global peak detector which stays at inh_{max} for the duration of the pulse, ensuring that the inhibitory signal robustly suppresses spiking activity for a wide range of potential output pulse widths.

As with the single neuron output rule, the single neuron threshold adaptation rule of Equation 4 can be modified to Equation 9 (underlined terms added) to utilize the global inhibitory signal for the multi-neuron case. This modification prevents a neuron's threshold being affected by the presentation of patterns that another neuron is better adapted to. The addition of the underlined terms in the first line of Equation 9 means that a neuron's threshold can only rise when its membrane potential exceeds its threshold and the inhibitory signal is not already active, or if the neuron itself spiked in the previous time step. The fall in the threshold is similarly conditioned on the neuron having spiked *before* the global inhibitory signal was activated, such that only the very best adapted neuron, i.e., the one that generated the inhibitory signal in the first place, adapts its threshold.

$$\Theta_n(t) = \begin{cases} \Theta(t-1) + \Theta_{rise} & \text{if } \sum_i r_i(t) > \Theta(t-1) \\ & \underline{\wedge\ (inh(t-1) = 0 \vee s_n(t-1) = 1)} \\ \Theta(t-1) - \Theta_{fall} & \text{if } \left(\sum_i r_i(t) = 0\ \wedge\ \sum_i r_i(t-1) > 0\right. \\ & \underline{\wedge\ inh(t-1) = 0)} \\ & \underline{\vee\ (s(t) = 0 \wedge s(t-1) = 1)} \\ \Theta(t-1) & \text{else} \end{cases} \quad (9)$$

Such a global inhibitory signal has been utilized in LIF neurons (Afshar et al., 2012; Tapson and van Schaik, 2012) and synaptic weight STDP neurons as a means of decorrelating neuronal firing patterns (Masquelier et al., 2009; Habenschuss et al., 2013). Here, however, its use is subtly different from both. Although in LIF and synaptic STDP architectures and in SKAN a global inhibitory signal results in the decorrelation of output spikes, in the purely synaptic weight adapting schemes the neuron's response time remains static and does not improve with adaptation and in the LIF networks (Afshar et al., 2012; Tapson and van Schaik, 2012) there is no lasting adaptation at all. SKAN's improved response time due to kernel adaptation and the global inhibitory signal realize a positive feedback mechanism absent in previous models. In a SKAN network a neuron's small initial advantage for a pattern results in a slightly earlier output spike. This output spike globally inhibits all other neurons, which in turn results in exclusive adaptation to the pattern by the first spiking neuron. This further improves that neuron's response time for the pattern and increases the likelihood of the neuron being the first to spike due to a subsequent presentation of the same pattern, even in the presence of temporal jitter. Thus, the adaptation of SKAN's kernels and thresholds, together with the global inhibitory network, mean that the neuron whose initial state is closest to the presented pattern will be the first to respond and prevent all other neurons adapting to this pattern. This effectively "hides" the pattern from the other neurons and allows unsupervised spike pattern classification by the network as whole as demonstrated in the proceeding Results Sections.

There are two important constraints adhered to by the preceding modification of the SKAN rules. The first constraint is that the required connectivity does not increase combinatorially with the number of neurons as described in Equation 10 since the only feedback path is from the single global inhibitory signal.

$$\text{total connections} = (\text{number of input channel} + 2) \times \text{number of neurons} \quad (10)$$

The second constraint is that no complex central controller is required for arbitration between the neurons. In competitive neural network schemes where a neuron's fitness is expressed as a real value from each neuron to a Winner-Take-All network, multiple bits (connections in hardware) are required to transport this information. Alternatively rate based systems encode such real valued signal over time in their spike rate which are then utilized by a corresponding rate based Winner-Take-All system. But in SKAN these requirements are reduced. Since a neuron's latencies correlates with its adaptation to a target pattern, the neurons do not need to report a real value but only a single bit. This mode of operation can be interpreted as either a connectivity saving or as a speed saving with respect to alternative multi-bit or rate based systems respectively. Furthermore, because of the robustness of the system, checking for, or prevention of, simultaneous output spikes is not necessary. Random initial heterogeneities in the neurons' parameters and/or noise in their signals is enough to eliminate the need for central control by pushing the neurons away from input space saddle points toward their stable non-overlapping receptive fields.

RESULTS

ONLINE UNSUPERVISED SPATIO-TEMPORAL SPIKE PATTERN CLASSIFICATION

In the following sections the classification performance of SKAN is tested in several ways. For these tests equally likely spatio-temporal spike patterns, each with one spike per channel per presentation were presented in random sequences to the SKAN network. **Table 1** details the parameters used in all the tests. These parameters were deliberately chosen for non-optimized performance so as to try to mimic the use of the system in the wild by a non-expert user. Examples of available optimizations include: higher *ddr* values which result in faster converging systems, reduced $\Theta_{rise}/\Theta_{fall}$ ratio for improved robustness to noise, increased $\Delta r_{max}/\Delta r_{i,n}(t=0)$ ratio and increased pattern widths for enhanced pattern selectivity.

Hardware efficiency through 1-to-1 neuron to pattern allocation at the local level

Through temporal competition a local network of mutually inhibiting SKANs can efficiently distribute limited neural resources in a hardware implementation to observed spatio-temporal patterns as is demonstrated in **Figure 14**.

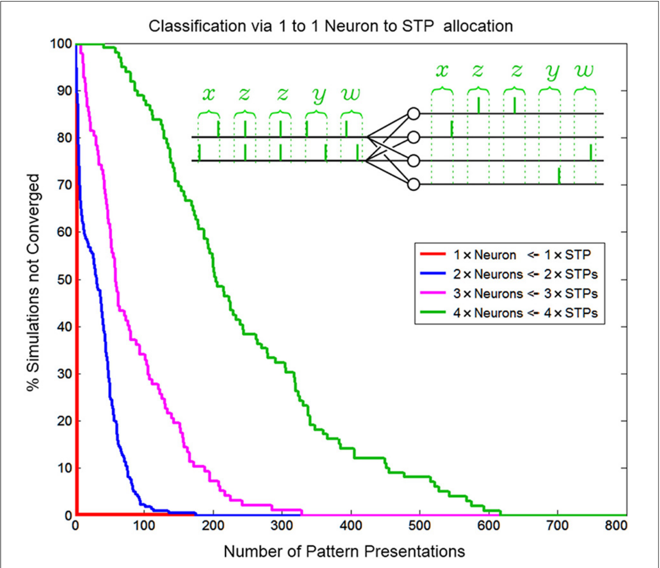


FIGURE 14 | Convergence rate of as a function of neuron/pattern numbers and number of pattern presentations for a 1-to-1 two input neuron to pattern allocating network. As the number of patterns/neurons increases the system requires longer pattern sequences to correctly allocate exactly one unique pattern to each neuron. The inset shows the five consecutive correct classifications of four patterns by four neurons.

Table 1 | Parameter values used for all results.

Parameter	Value	Description
ddr	1	Change in the kernel step size. Higher value results in faster adaptation; lower values are more robust to noise
$w(r_{max})$	10,000	Maximum kernel height (synaptic weight)
r_{min}	0	The kernel signal $r(t)$ saturates at zero
$\Delta r_{i,n}(t=0)$	100 × (1 + rand)	Initial kernel step size (For each input *i* to each neuron *n*) The randomized initialization allows different neurons to learn different patterns
Δr_{max}	400	Maximum kernel step size
$r_{i,n}(t=0)$	0	Initial kernel value
Θ_{rise}	40 × *inputs*	Rise in threshold during output spike, where *inputs* is the number of input channels per neuron
Θ_{fall}	100 × *inputs*	Fall in threshold due to input spikes, where *inputs* is the number of input channels per neuron
inh_{max}	100	Initial value of the inhibitory countdown
inh_{decay}	1	Step size of the inhibitory countdown. As a rule of thumb use: $inh_{max}/inh_{decay} = \min[\Delta r_{i,n}\ (t=0)]$
T	400 Δ*t*	Time between pattern presentations

Similar to biological systems, in a SKAN network there is no supervisor switching the network from a training mode to a testing mode so there is no distinction between learning and recognition. This means that attempting to test SKANs in the traditional neural network sense by switching off a network's adaptation mechanisms would disable the system. Thus, to test the network's performance 1000 simulations were generated for each instance of the network, with up to 800 pattern presentations each. The network was considered to have converged to a stable solution when 20 consecutive patterns were correctly classified by the network, i.e., with a single neuron responding per spatiotemporal pattern. This is illustrated in the inset of **Figure 14**. Correct classification was defined as the case where a neuron spikes if and only if its target pattern is presented and where the neurons consistently spike for the same learnt target pattern. Also, a single neuron should spike once for each input pattern and no extra output spikes occur. The percentage of simulations that had not converged to correct classification was recorded as a function of the number of patterns presented, and is shown in **Figure 14**. Simulations were terminated once a network had converged. The number of consecutive patterns was chosen as20 to reduce the likelihood that the observed "correct" response of the network was due to chance.

Classification performance as a function of spatio-temporal pattern dimension

The problem of coordinating multiple synapses for unsupervised neuronal classification in SNN models, whether through simply learning synaptic weights or through more complex pathways, is difficult (Jimenez Rezende and Gerstner, 2014). In SKAN the hybrid synapto-dendritic kernel adaptation produces convergence profiles shown in **Figure 15**. These results show how the convergence profiles of SKAN change with the number of active input channels. Additionally, the right panel in **Figure 15** shows the effect of increasing the resolution of the spatio-temporal pattern. Doubling the number of time steps in the maximal width of the target pattern *PW*, results in improved convergence profiles.

Classification in the presence of temporal noise

In order for SKAN to operate as an effective classifier competing neurons must balance the requirements of selectivity and generalization. In the spatio-temporal context, generalization takes the form of temporal jitter noise. In this context neurons must recognize patterns closest to their learnt target pattern despite the presence of temporal noise, while not recognizing other similarly noise corrupted patterns that are closer to the target patterns learnt by other neurons. Furthermore, the neurons should not expect the learning phase to be any less noisy than the testing phase or even for there to be any such distinct separation between learning and recognition. As well the neurons should maintain their correct learning and recognition behavior across a wide range of noise levels and they should ideally do so without the requirement for external adjustment of their parameters. SKAN satisfies all these requirements. The classification performance of SKAN is robust to temporal jitter noise as illustrated in **Figure 16** where two neurons act as two Kalman filters with shared inputs attempting to learn the statistics of two noisy but distinct ISI generating processes.

Because of the constant adaptation of the neurons, moderate levels of temporal noise with standard deviation up to $\sigma = 0.25$ Δt, which is 1/80th of the pattern width, either do not affect or actually *improve* SKAN performance. With high temporal noise levels, i.e., with a standard deviation that is 1/20th the width of the pattern ($\sigma = 1\ \Delta t$), the convergence profile is still similar to that of the noiseless case. Such levels of temporal noise can disable a conventional processor and even some neural networks. Even at the

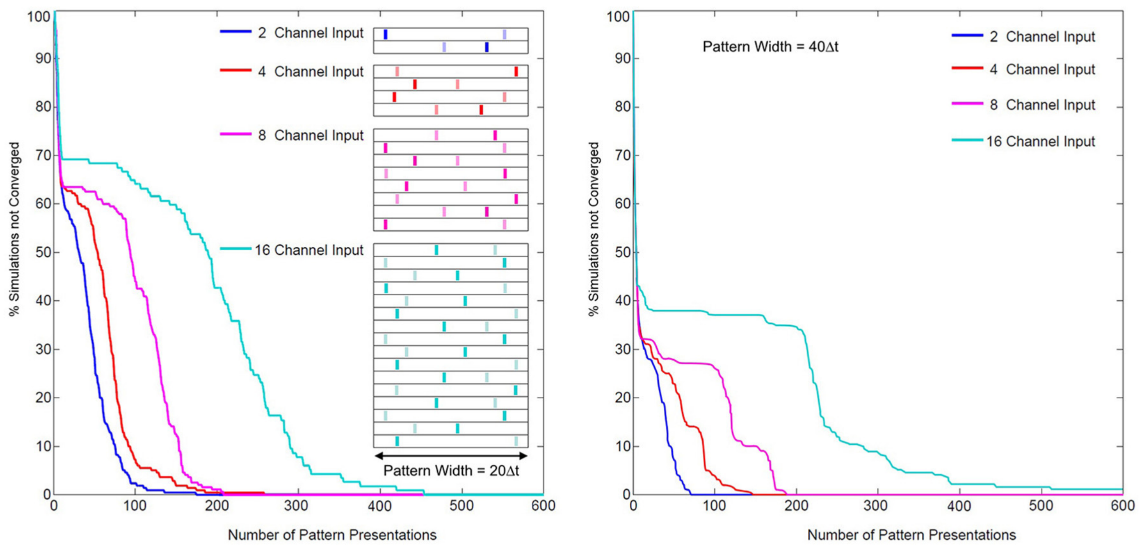


FIGURE 15 | Convergence rates as a function of input channel dimension and pattern width. Left panel: two random target patterns (light and dark bars) of maximal pattern width $PW = 20\ \Delta t$ and of dimensions 2, 4, 8, and 16, were presented at random to a two neuron network, with the convergence of simulations plotted over the number of presentations. **Right panel:** the same test with maximal pattern width $PW = 40\ \Delta t$.

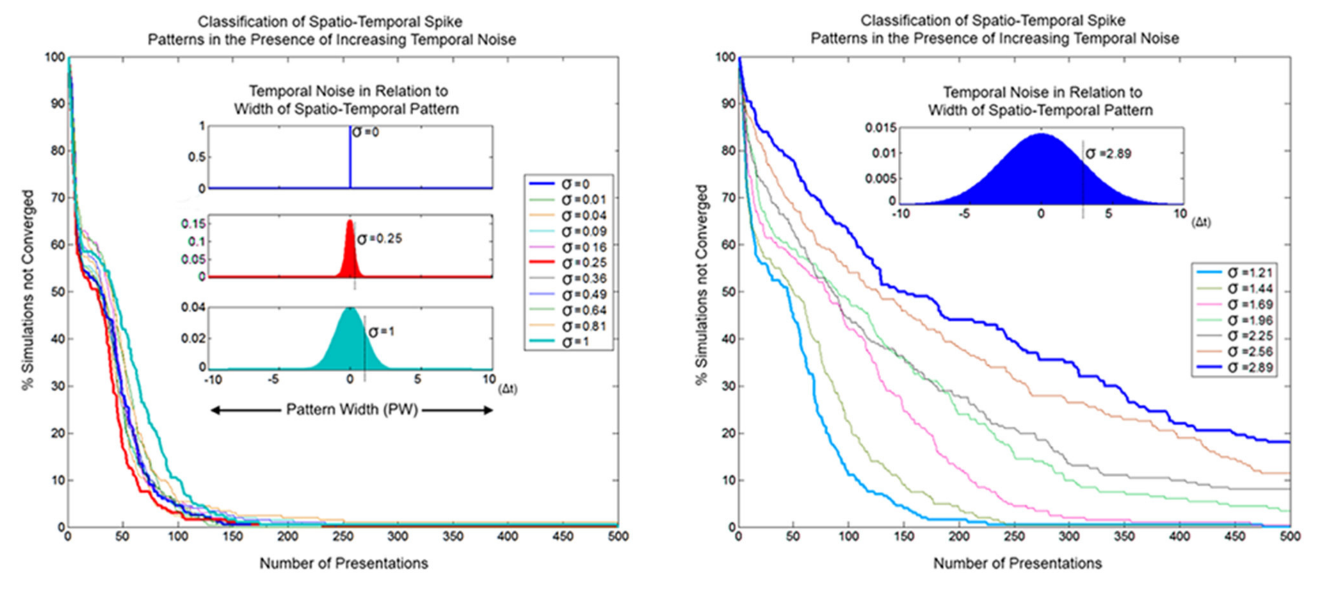


FIGURE 16 | Convergence as a function of temporal noise and pattern presentations in a two neuron network with two input channels. The insets illustrate selected noise distributions relative to the maximal pattern width (*PW* = 20 Δ*t*). The left panel shows convergence profiles due to temporal noise distribution with standard deviation σ = 0–1 Δ*t*. At lower noise levels the convergence profile is approximately the same or faster (red) than the zero noise case. The right panel shows the same for σ ≈ 1–3 Δ*t*.

extreme, with noise that has a standard deviation more than 1/7th of the pattern width, some simulations still result in the neurons correctly classifying the separate ISI sources.

As a temporal coding scheme, the robustness of SKAN's learning algorithm to temporal noise is critical for potential real-world applications, where the ability to operate (and degrade gracefully) in noisy, dynamic environments is favored over ideal performance in ideal noise free circumstances.

IMPLEMENTATION IN FPGA

SKAN was implemented in an Altera Cyclone-V GX FPGA, a low-end FPGA containing 77,000 programmable logic elements (LEs). The functions of SKAN were programmed based on the equations described in the earlier sections, written using the Verilog hardware description language, with no optimization techniques employed. A key feature of this design is that no multipliers are required: SKAN is executed entirely using simple summation and logical operations only, thus significantly reducing computational complexity and hardware resources. Registers are used to store required design parameters. **Table 2** shows the utilization of the FPGA in terms of registers, adaptive logic modules (ALMs) and the percentage of resources used for SKAN modules containing different number of synapses. From this we can see that SKAN is efficient in its usage of hardware resources. Results from the FPGA are identical to the simulated results as integers were used for both the simulations and the FPGA and therefore no approximations were required. Integers were used to avoid floating point operations, thereby reducing computation. An efficient use of hardware resources, reduced computational effort, and its ease of implementation make SKAN an attractive neuromorphic solution in terms of both cost and performance.

Table 2 | Altera Cyclone V FPGA resource usage for a SKAN neuron with different number of synapses.

	FPGA resource usage		
No. of synapses	Single bit registers	Adaptive logic modules (ALMs)*	Usage percentage (%)
1	48	189	0.6
2	72	297	1.0
4	121	501	1.7
8	218	922	3.2
16	411	1580	5.4

**An ALM is equivalent to 2.65 logic elements (LEs).*

DISCUSSION

As outlined in the introduction a limiting factor in many neuromorphic systems is the large number of complex synapses which require multipliers and high connectivity networks required for robust performance. A simple solution to this challenge has been to physically implement of one or a few instances of these complex elements and use time multiplexing and AER to generate larger virtual networks. The kernels of SKAN which do not require multipliers allow more synapses to be physically realized in hardware while their adaptability means that better performance can be achieved using fewer synapses. Furthermore, the time based operation of the neurons reduces the required connectivity. This potentially allows entire networks to be physically implemented in hardware. Such small or medium sized networks whose behaviors have been described in this report can then be cascaded or multiplexed to form larger networks. Such solutions could potentially occupy a middle ground between fully hardware implemented networks with limited connectivity but high bandwidth and single neuron realizations with high connectivity and limited operating speeds. While the focus of this introductory report is on characterization of small non-optimized SKAN networks, preliminary work on the application of the architecture to larger, more difficult recognition tasks such as unsupervised learning of the MNIST dataset has not revealed any limits to the capabilities of larger, more optimized networks. Future work will focus on comparison of SKAN networks to other neural network solutions on established datasets, comparison of the inference capabilities of the neuron to optimal probabilistic estimators and the investigation of the combined effects of adaptation of SKAN's kernels and the adaptation of its synaptic weight parameter *w* which allows encoding of synaptic signal to noise ratios for each input channel.

CONCLUSION

In this paper we have presented the SKAN, a neuromorphic implementation of a spiking neuron that performs statistical inference and unsupervised learning and spatio-temporal spike pattern classification. The use of simple adaptable kernels was shown to represent an efficient solution to hardware realized neural networks without the need for multipliers while SKAN operation was shown to be robust in the presence of noise allowing potential applications in noisy real-world environments. Finally it was shown that SKAN is hardware efficient and easily implemented on an FPGA.

REFERENCES

Afshar, S., Cohen, G. K., Wang, R. M., van Schaik, A., Tapson, J., Lehmann, T., et al. (2013). The ripple pond: enabling spiking networks to see. *Front. Neurosci.* 7:212. doi: 10.3389/fnins.2013.00212

Afshar, S., Kavehei, O., van Schaik, A., Tapson, J., Skafidas, S., and Hamilton, T. J. (2012). "Emergence of competitive control in a memristor-based neuromorphic circuit," in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (Brisbane, QLD), 1–8.

Arthur, J. V., Merolla, A., Akopyan, F., Alvarez, R., Cassidy, A., Chandra, S., et al. (2012). "Building block of a programmable neuromorphic substrate: a digital neurosynaptic core," in *Proceedings of the International Joint Conference on Neural Networks* (Brisbane, QLD).

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., and Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron* 76, 695–711. doi: 10.1016/j.neuron.2012.10.038

Beyeler, M., Dutt, N. D., and Krichmar, J. L. (2013). Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule. *Neural Netw.* 48, 109–124. doi: 10.1016/j.neunet.2013.07.012

Bhatt, D. H., Zhang, S., and Gan, W. B. (2009). Dendritic spine dynamics. *Annu. Rev. Physiol.* 71, 261–282. doi: 10.1146/annurev.physiol.010908.163140

Boerlin, M., Machens, C. K., and Denève, S. (2013). Predictive coding of dynamical variables in balanced spiking networks. *PLoS Comput. Biol.* 9:e1003258. doi: 10.1371/journal.pcbi.1003258

Brunel, N., Hakim, V., and Richardson, M. J. E. (2014). Single neuron dynamics and computation. *Curr. Opin. Neurobiol.* 25, 149–155. doi: 10.1016/j.conb.2014.01.005

Bullmore, E. T., and Sporns, O. (2012). The economy of brain network organization. *Nat. Rev. Neurosci.* 13, 336–349. doi: 10.1038/nrn3214

Chicca, E., Stefanini, F., Bartolozzi, C., and Indiveri, G. (2014). Neuromorphic electronic circuits for building autonomous cognitive systems. *Proc. IEEE* 99, 1–22. doi: 10.1109/JPROC.2014.2313954

Colgan, L., and Yasuda, R. (2014). Plasticity of dendritic spines: subcompartmentalization of signaling. *Annu. Rev. Physiol.* 76, 365–385. doi: 10.1146/annurev-physiol-021113-170400

Corneil, D. S., Neftci, E., Indiveri, G., and Pfeiffer, M. (2014). *Learning, Inference, and Replay of Hidden State Sequences in Recurrent Spiking Neural Networks.* Salt Lake City, UT.

Denève, S. (2008). Bayesian spiking neurons I: inference. *Neural Comput.* 20, 91–117. doi: 10.1162/neco.2008.20.1.91

Dowrick, T., Hall, S., and McDaid, L. (2013). A simple programmable axonal delay scheme for spiking neural networks. *Neurocomputing* 108, 79–83. doi: 10.1016/j.neucom.2012.12.004

Fontaine, B., Peña, J. L., and Brette, R. (2014). Spike-threshold adaptation predicted by membrane potential dynamics *in vivo*. *PLoS Comput. Biol.* 10:e1003560. doi: 10.1371/journal.pcbi.1003560

George, S., Hasler, J., Koziol, S., Nease, S., and Ramakrishnan, S. (2013). Low power dendritic computation for wordspotting. *J. Low Power Electron. Appl.* 3, 73–98. doi: 10.3390/jlpea3020073

Gupta, A., and Long, L. N. (2009). Hebbian learning with winner take all for spiking neural networks. *Neural Netw.* 81, 1054–1060. doi: 10.1109/IJCNN.2009.5178751

Gütig, R. (2014). To spike, or when to spike? *Curr. Opin. Neurobiol.* 25, 134–139. doi: 10.1016/j.conb.2014.01.004

Habenschuss, S., Puhr, H., and Maass, W. (2013). Emergence of optimal decoding of population codes through STDP. *Neural Comput.* 25, 1371–1407. doi: 10.1162/NECO_a_00446

Hamilton, T. J., Afshar, S., van Schaik, A., and Tapson, J. (2014). Stochastic electronics: a neuro-inspired design paradigm for integrated circuits. *Proc. IEEE* 102, 843–859. doi: 10.1109/JPROC.2014.2310713

Hamilton, T. J., and Tapson, J. (2011). "A neuromorphic cross-correlation chip," in *Proceedings - IEEE International Symposium on Circuits and Systems* (Rio de Janeiro), 865–868.

Harnett, M. T., Makara, J. K., Spruston, N., Kath, W. L., and Magee, J. C. (2012). Synaptic amplification by dendritic spines enhances input cooperativity. *Nature* 491, 599–602. doi: 10.1038/nature11554

Herz, A. V. M., Gollisch, T., Machens, C. K., and Jaeger, D. (2006). Modeling single-neuron dynamics and computations: a balance of detail and abstraction. *Science* 314, 80–85. doi: 10.1126/science.1127240

Hsu, C. C., Parker, A. C., and Joshi, J. (2010). "Dendritic computations, dendritic spiking and dendritic plasticity in nanoelectronic neurons," in *Midwest Symposium on Circuits and Systems* (Seattle, WA), 89–92.

Hunsberger, E., Scott, M., and Eliasmith, C. (2014). The competing benefits of noise and heterogeneity in neural coding. *Neural Comput.* 26, 1600–1623. doi: 10.1162/NECO_a_00621

Hussain, S., Basu, A., Wang, R. M., and Julia Hamilton, T. (2014). Delay learning architectures for memory and classification. *Neurocomputing* 138, 14–26. doi: 10.1016/j.neucom.2013.09.052

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., and Prodromakis, T. (2013). Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology* 24:384010. doi: 10.1088/0957-4484/24/38/384010

Izhikevich, E. M. (2006). Polychronization: computation with spikes. *Neural Comput.* 18, 245–282. doi: 10.1162/089976606775093882

Jaeger H. (2001). *The Echo State Approach to Analyzing and Training Recurrent Neural Networks.* Technical Report, GMD Report 148, GMD-German National Research Institute for Computer Science.

Jimenez Rezende, D., and Gerstner, W. (2014). Stochastic variational learning in recurrent spiking networks. *Front. Comput. Neurosci.* 8:38. doi: 10.3389/fncom.2014.00038

Kappel, D., Nessler, B., and Maass, W. (2014). STDP installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLoS Comput. Biol.* 10:e1003511. doi: 10.1371/journal.pcbi.1003511

Kasabov, N., Dhoble, K., Nuntalid, N., and Indiveri, G. (2013). Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural Netw.* 41, 188–201. doi: 10.1016/j.neunet.2012.11.014

Kasai, H., Fukuda, M., Watanabe, S., Hayashi-Takagi, A., and Noguchi, J. (2010a). Structural dynamics of dendritic spines in memory and cognition. *Trends Neurosci.* 33, 121–129. doi: 10.1016/j.tins.2010.01.001

Kasai, H., Hayama, T., Ishikawa, M., Watanabe, S., Yagishita, S., and Noguchi, J. (2010b). Learning rules and persistence of dendritic spines. *Eur. J. Neurosci.* 32, 241–249. doi: 10.1111/j.1460-9568.2010.07344.x

Kim, J. K., Knag, P., Chen, T., and Zhang, Z. (2014). Efficient hardware architecture for sparse coding. *IEEE Trans. Signal Process.* 62, 4173–4186. doi: 10.1109/TSP.2014.2333556

Kuhlmann, L., Hauser-Raspe, M., Manton, J. H., Grayden, D. B., Tapson, J., and van Schaik, A. (2014). Approximate, computationally efficient online learning in Bayesian spiking neurons. *Neural Comput.* 26, 472–496. doi: 10.1162/NECO_a_00560

Laughlin, S. B. (2001). Energy as a constraint on the coding and processing of sensory information. *Curr. Opin. Neurobiol.* 11, 475–480. doi: 10.1016/S0959-4388(00)00237-3

Lee, K. F. H., Soares, C., and Béïque, J. C. (2012). Examining form and function of dendritic spines. *Neural Plasticity* 2012:704103. doi: 10.1155/2012/704103

Lee, K. F. H., Soares, C., and Béïque, J.-C. (2014). Tuning into diversity of homeostatic synaptic plasticity. *Neuropharmacology* 78, 31–37. doi: 10.1016/j.neuropharm.2013.03.016

Levy, W. B., and Baxter, R. A. (1996). Energy efficient neural codes. *Neural Comput.* 8, 531–543. doi: 10.1162/neco.1996.8.3.531

Llinas, R. (2008). Neuron. *Scholarpedia* 3:1490. doi: 10.4249/scholarpedia.1490

Losonczy, A., Makara, J. K., and Magee, J. C. (2008). Compartmentalized dendritic plasticity and input feature storage in neurons. *Nature* 452, 436–441. doi: 10.1038/nature06725

Maass, W. (2014). Noise as a resource for computation and learning in networks of spiking neurons. *Proc. IEEE* 102, 860–880. doi: 10.1109/JPROC.2014.2310593

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* 14, 2531–2560. doi: 10.1162/089976602760407955

Marr, B., and Hasler, J. (2014). Compiling probabilistic, bio-inspired circuits on a field programmable analog array. *Front. Neurosci.* 8:86. doi: 10.3389/fnins.2014.00086

Masquelier, T., Guyonneau, R., and Thorpe, S. J. (2009). Competitive STDP-based spike pattern learning. *Neural Comput.* 21, 1259–1276. doi: 10.1162/neco.2008.06-08-804

McDonnell, M. D., and Ward, L. M. (2011). The benefits of noise in neural systems: bridging theory and experiment. *Nat. Rev. Neurosci.* 12, 415–426. doi: 10.1038/nrn3061

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., and Modha, D. S. (2011). "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," in *Proceedings of the Custom Integrated Circuits Conference* (San Jose, CA).

Misra, J., and Saha, I. (2010). Artificial neural networks in hardware: a survey of two decades of progress. *Neurocomputing* 74, 239–255. doi: 10.1016/j.neucom.2010.03.021

Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Circuits Syst.* 3, 32–42. doi: 10.1109/TBCAS.2008.2005781

Nessler, B., Pfeiffer, M., Buesing, L., and Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through Spike-Timing-Dependent Plasticity. *PLoS Comput. Biol.* 9:e1003037. doi: 10.1371/journal.pcbi.1003037

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* 7:178. doi: 10.3389/fnins.2013.00178

Papoutsi, A., Kastellakis, G., Psarrou, M., Anastasakis, S., and Poirazi, P. (2014). Coding and decoding with dendrites. *J. Physiol. Paris* 108, 18–27. doi: 10.1016/j.jphysparis.2013.05.003

Paulin, M. G., and van Schaik, A. (2014). *Bayesian Inference with Spiking Neurons*. arXiv preprint arXiv:1406.5115

Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., et al. (2012). Is a 4-bit synaptic weight resolution enough? – constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci.* 6:90. doi: 10.3389/fnins.2012.00090

Pouget, A., Beck, J. M., Ma, W. J., and Latham, P. E. (2013). Probabilistic brains: knowns and unknowns. *Nat. Neurosci.* 16, 1170–1178. doi: 10.1038/nn.3495

Rahimi Azghadi, M., Iannella, N., Al-Sarawi, S. F., Indiveri, G., and Abbott, D. (2014). Spike-based synaptic plasticity in silicon: design, implementation, application, and challenges. *Proc. IEEE* 102, 717–737. doi: 10.1109/JPROC.2014.2314454

Ramakrishnan, S., Wunderlich, R., Hasler, J., and George, S. (2013). Neuron array with plastic synapses and programmable dendrites. *IEEE Trans. Biomed. Circuits Syst.* 7, 631–642. doi: 10.1109/TBCAS.2013.2282616

Rochefort, N. L., and Konnerth, A. (2012). Dendritic spines: from structure to *in vivo* function. *EMBO Rep.* 13, 699–708. doi: 10.1038/embor.2012.102

Scholze, S. (2011). VLSI implementation of a 2.8 Gevent/s packet-based AER interface with routing and event sorting functionality. *Front. Neurosci.* 5:117. doi: 10.3389/fnins.2011.00117

Seo, J. S., Brezzo, B., Liu, Y., Parker, B. D., Esser, S. K., Montoye, R. K., et al. (2011). "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proceedings of the Custom Integrated Circuits Conference* (San Jose, CA).

Serrano-Gotarredona, T., Prodromakis, T., and Linares-Barranco, B. (2013). A proposal for hybrid memristor-CMOS spiking neuromorphic learning systems. *IEEE Cir. Syst. Mag.* 13, 74–88. doi: 10.1109/MCAS.2013.2256271

Shah, M. M., Hammond, R. S., and Hoffman, D. A. (2010). Dendritic ion channel trafficking and plasticity. *Trends Neurosci.* 33, 307–316. doi: 10.1016/j.tins.2010.03.002

Sheik, S., Chicca, E., and Indiveri, G. (2012). "Exploiting device mismatch in neuromorphic VLSI systems to implement axonal delays," in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (Brisbane), 1–6.

Sheik, S., Pfeiffer, M., Stefanini, F., and Indiveri, G. (2013). "Spatio-temporal spike pattern classification in neuromorphic systems," in *Biomimetic and Biohybrid Systems*, eds N. F. Lepora, A. Mura, H. G. Krapp, P. F. M. J. Verschure, and T. J. Prescott (Heidelberg: Springer), 262–273. doi: 10.1007/978-3-642-39802-5_23

Silver, R. A. (2010). Neuronal arithmetic. *Nat. Rev. Neurosci.* 11, 474–489. doi: 10.1038/nrn2864

Sjöström, P. J., Rancz, E. A., Roth, A., and Häusser, M. (2008). Dendritic excitability and synaptic plasticity. *Physiol. Rev.* 88, 769–840. doi: 10.1152/physrev.00016.2007

Smith, S. L., Smith, I. T., Branco, T., and Häusser, M. (2013). Dendritic spikes enhance stimulus selectivity in cortical neurons *in vivo*. *Nature* 503, 115–120. doi: 10.1038/nature12600

Sofatzis, R. J., Afshar, S., and Hamilton, T. J. (2014a). "The synaptic kernel adaptation network," in *The Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)* (Melbourne, VIC).

Sofatzis, R. J., Afshar, S., and Hamilton, T. J. (2014b). "Rotationally invariant vision recognition with neuromorphic transformation and learning networks," in *The Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)* (Melbourne, VIC).

Spruston, N. (2008). Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.* 9, 206–221. doi: 10.1038/nrn2286

Tapson, J. C., Cohen, G. K., Afshar, S., Stiefel, K. M., Buskila, Y., Wang, R. M., et al. (2013). Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Front. Neurosci.* 7:153. doi: 10.3389/fnins.2013.00153

Tapson, J., and van Schaik, A. (2012). "An asynchronous parallel neuromorphic ADC architecture," in *2012 IEEE International Symposium on Circuits and Systems* (Seoul), 2409–2412. doi: 10.1109/ISCAS.2012.6271783

Topol, A. W., La Tulipe, D. C., Shi, L., Frank, D. J., Bernstein, K., Steen, S. E., et al. (2006). Three-dimensional integrated circuits. *IBM J. Res. Dev*. 50, 491–506. doi: 10.1147/rd.504.0491

Tully, P. J., Hennig, M. H., and Lansner, A. (2014). Synaptic and nonsynaptic plasticity approximating probabilistic inference. *Front. Synaptic Neurosci.* 6:8. doi: 10.3389/fnsyn.2014.00008

Van Rullen, R., and Thorpe, S. J. (2001). Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Comput.* 13, 1255–1283. doi: 10.1162/08997660152002852

Wang, R. M., Hamilton, T. J., Tapson, J. C., and van Schaik, A. (2014). A mixed-signal implementation of a polychronous spiking neural network with delay adaptation. *Front. Neurosci.* 8:51. doi: 10.3389/fnins.2014.00051

Wang, Y., and Liu, S. C. (2013). Active processing of spatio-temporal input patterns in silicon dendrites. *IEEE Trans. Biomed. Circuits Syst.* 7, 307–318. doi: 10.1109/TBCAS.2012.2199487

Xu, W., Morishita, W., Buckmaster, S., Pang, Z. P., Malenka, R. C., and Südhof, T. C. (2012). Distinct neuronal coding schemes in memory revealed by selective erasure of fast synchronous synaptic transmission. *Neuron* 73, 990–1001. doi: 10.1016/j.neuron.2011.12.036

Yegnanarayana, B. (1999). *Artificial Neural Networks*. New Delhi: Prentice-Hall India.

Yoshihara, Y., De Roo, M., and Muller, D. (2009). Dendritic spine formation and stabilization. *Curr. Opin. Neurobiol.* 19, 146–153. doi: 10.1016/j.conb.2009.05.013

Yu, Y., and Lee, T. (2003). Dynamical mechanisms underlying contrast gain control in single neurons. *Phys. Rev. E* 68:011901. doi: 10.1103/PhysRevE.68.011901

Zamarreno-Ramos, C., Linares-Barranco, A., Serrano-Gotarredona, T., and Linares-Barranco, B. (2013). Multicasting mesh AER: a scalable assembly approach for reconfigurable neuromorphic structured AER systems. Application to ConvNets. *IEEE Trans. Biomed. Circuits Syst.* 7, 82–102. doi: 10.1109/TBCAS.2012.2195725

Zhu, J., and Sutton, P. (2003). FPGA implementations of neural networks - A survey of a decade of progress. *Lect. Notes Comput. Sci.* 2778, 1062–1066. doi: 10.1007/978-3-540-45234-8_120

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 14 August 2014; accepted: 05 November 2014; published online: 25 November 2014.

Citation: Afshar S, George L, Tapson J, van Schaik A and Hamilton TJ (2014) Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels. Front. Neurosci. ***8****:377. doi: 10.3389/fnins.2014.00377*

This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.

ORIGINAL RESEARCH ARTICLE
published: 24 February 2015
doi: 10.3389/fnins.2015.00046

Spatiotemporal features for asynchronous event-based data

Xavier Lagorce[1]*, Sio-Hoi Ieng[1], Xavier Clady[1], Michael Pfeiffer[2] and Ryad B. Benosman[1]

[1] *Equipe de Vision et Calcul Naturel, UMR S968 Institut National de la Santé et de la Recherche Médicale, Centre National de la Recherche Scientifique UMR 7210, Centre Hospitalier National d' Ophtalmologie des Quinze-Vingts, Université Pierre et Marie Curie, Paris, France*
[2] *Institute of Neuroinformatics, University of Zürich and Eidgenössische Technische Hochschule (ETH) Zürich, Zürich, Switzerland*

Edited by:
Elisabetta Chicca, University of Bielefeld, Germany

Reviewed by:
Dylan Richard Muir, University of Basel, Switzerland
Mattia Rigotti, IBM Research, USA
Bernabe Linares-Barranco, Instituto de Microelectrónica de Sevilla, Spain
Federico Corradi, Institute of Neuroinformatics, University of Zürich and ETH Zürich, Switzerland

****Correspondence:***
Xavier Lagorce, Equipe de Vision et Calcul Naturel, UMR S968 Institut National de la Santé et de la Recherche Médicale, Centre National de la Recherche Scientifique UMR 7210, Centre Hospitalier National d' Ophtalmologie des Quinze-Vingts, Université Pierre et Marie Curie, 17 Rue Moreau, 75012 Paris, France
e-mail: xavier.lagorce@upmc.fr

Bio-inspired asynchronous event-based vision sensors are currently introducing a paradigm shift in visual information processing. These new sensors rely on a stimulus-driven principle of light acquisition similar to biological retinas. They are event-driven and fully asynchronous, thereby reducing redundancy and encoding exact times of input signal changes, leading to a very precise temporal resolution. Approaches for higher-level computer vision often rely on the reliable detection of features in visual frames, but similar definitions of features for the novel dynamic and event-based visual input representation of silicon retinas have so far been lacking. This article addresses the problem of learning and recognizing features for event-based vision sensors, which capture properties of truly spatiotemporal volumes of sparse visual event information. A novel computational architecture for learning and encoding spatiotemporal features is introduced based on a set of predictive recurrent reservoir networks, competing via winner-take-all selection. Features are learned in an unsupervised manner from real-world input recorded with event-based vision sensors. It is shown that the networks in the architecture learn distinct and task-specific dynamic visual features, and can predict their trajectories over time.

Keywords: echo-state networks, spatiotemporal, feature extraction, recognition, silicon retinas

1. INTRODUCTION

Humans learn efficient strategies for visual perception tasks by adapting to their environment through interaction, and recognizing salient features. In contrast, most current computer vision systems have no such learning capabilities. Despite the accumulated evidence of visual feature learning in humans, little is known about the mechanisms of visual learning (Wallis and Bülthoff, 1999). A fundamental question in the study of visual processing is the problem of feature selection: which features of a scene are extracted and represented by the visual cortex? Classical studies of feature selectivity of cortical neurons have linked neural responses to properties of local patches within still images (Hubel and Wiesel, 1962; Olshausen and Field, 1997). Conventional artificial vision systems rely on sampled acquisition that acquires static snapshots of the scene at fixed time intervals. This regular sampling of visual information imposes an artificial timing for events detected in a natural scene. One of the main drawbacks of representing a natural visual scene through a collection of snapshot images is the complete lack of dynamics and the high amount of redundancy in the acquired data. Every pixel is sampled continuously, even if its output value remains unchanged. The output of a pixel is then unnecessarily digitized, transmitted, stored, and processed, even if it does not provide any new information that was not available in preceding frames. This highly inefficient use of resources introduces severe limitations in computer vision applications, since the largely redundant acquired information lead to a waste of energy for acquisition, compression, decompression and processing (Lichtsteiner et al., 2008).

Biological observations confirm that still images are largely unknown to the visual system. Instead, biological sensory systems are massively parallel and data-driven (Gollisch and Meister, 2008). Biological retinas encode visual data asynchronously through sparse firing spike trains, rather than as frames of pixel values (Roska and Werblin, 2003). Current studies show that the visual system effortlessly combines the various features of visual stimuli to form coherent perceptual categories relying on a surprisingly high temporal resolution: the temporal offsets of on-bistratified retina cells responses show an average standard deviation of 3.5 ms (Berry et al., 1997; Uzzell and Chichilnisky, 2004). Neurons in the visual cortex also precisely follow the temporal dynamics of the stimuli up to a precision of 10 ms. In order to bridge the gap between artificial machine vision and biological visual perception, computational vision has taken inspiration from fundamental studies of visual mechanisms in animals (Hubel and Wiesel, 1962; Wallis and Rolls, 1997). One main focus of these approaches have been various computational models of simple and complex cells in the primary visual cortex (V1) Hubel and Wiesel (1962); Fukushima (1980); Riesenhuber and Poggio (1999), which are characterized by their preferred response to localized oriented bars. Typically, this orientation-tuned response of V1 cells has been modeled with Gabor Filters (Gabor, 1946), which have been used as the first layer of feature extraction for visual recognition tasks (Huang et al., 2004; Ilonen et al., 2007). The most well-known example of biologically inspired, although still frame-based model of object recognition is the HMAX model (Riesenhuber and Poggio, 1999; Serre et al., 2006; Mutch et al., 2010). It implements a feedforward neural network based on a first layer of Gabor filters followed by different layers realizing linear and non-linear operations modeled on primate cortex cells. However, HMAX like other approaches implementing neural networks to perform visual tasks (Lin and Huang, 2005) are still based on processing still images and therefore cannot capture key visual information mediated by time.

This paper introduces an unsupervised system that allows to extract visual spatiotemporal features from natural scenes. It does not rely on still images, but on the precise timing of spikes acquired by an asynchronous spike-based silicon retina (Lichtsteiner et al., 2008). The development of asynchronous event-based retinas has been initiated by the work of Mahowald and Mead (Mead and Mahowald, 1988). Neuromorphic asynchronous event-based retinas allow new insights into the capabilities of perceptual models to use time as a source of information. Currently available event-based vision sensors (Delbruck et al., 2010; Posch et al., 2011) produce compressed digital data in the form of time-stamped, localized events, thereby reducing latency and increasing temporal dynamic range compared to conventional imagers. Because pixel operation is now asynchronous and pixel circuits can be designed to have extremely high temporal resolution, silicon retinas accomplish both the reduction of over-sampling of highly redundant static information, as well as eliminating under-sampling of very fast scene dynamics, which in conventional cameras is caused by a fixed frame rate. Pixel acquisition and readout times of milliseconds to microseconds are achieved, resulting in temporal resolutions equivalent to conventional sensors running at tens to hundreds of thousands of frames per second, without the data overhead of conventional high-speed imaging. The implications of this approach for machine vision can hardly be overstated. Now, for the first time, the strict temporal resolution vs. data rate tradeoff that limits all frame-based vision acquisition can be overcome. Visual data acquisition simultaneously becomes fast and efficient. A recent review of these sensors can be found in Delbruck et al. (2010) and Posch et al. (2014).

Despite the efficiency of the sensor representation, it is far from straightforward to port methods that have proven successful in computer vision to the event-based vision domain. Much of the recent success of computer vision comes from the definition of robust and invariant feature or interest point extractors and descriptors (Lowe, 1999, 2004; Bay et al., 2008). Although such methods have proven to be very useful for static image classification, they require processing of the whole image, and do not take temporal information into account. Dynamical features for event data should instead recognize features only from novel visual input, and recognize them as they appear in the sparse input stream. This requires a model that can continuously process spiking inputs, and maintain a representation of the feature dynamics over time, even in the absence of input. Here we present an architecture for feature learning and extraction based on reservoir computing with recurrent neural networks (Schrauwen et al., 2007), which integrate event input from neuromorphic sensors, and compete via a Winner-Take-All (WTA) technique to specialize on distinct features by predicting their temporal evolution.

A proof of concept for the performance of the architecture is demonstrated in three experiments using natural recordings with event-based vision sensors. In the first experiment, we present a set of oriented bars to the camera in order to the show the capacity of the model to extract simple features in an unsupervised manner, using a big spatial receptive field to emphasize the graphical visualization of the learnt features. In the second experiment, the full capacity of the method is demonstrated by mapping the field of view to several small receptive fields, and showing that the model is still capable of reliably extracting features from the scene. The last experiment applies the architecture to complex object features. All experiments were conducted with real-world recordings from DVS cameras (Lichtsteiner et al., 2008), and thus are subject to the standard noise distribution of such sensors.

2. MATERIALS AND METHODS

2.1. EVENT-BASED ASYNCHRONOUS SENSORS

In our experiments we used asynchronous event-based input signals from a Dynamic Vision Sensor (DVS) (Lichtsteiner et al., 2008), which mimics the biological retina in silicon. It encodes visual information using the Address-Event Representation (AER), and has a spatial resolution of 128 × 128 pixels. The DVS outputs an asynchronous stream of events that signal local relative luminance changes in the scene, at the time they occur. Each pixel works independently for its receptive field, and creates events whenever the local luminance change since the time of the last emitted event exceeds a given threshold ΔI on a logarithmic scale. The typical threshold is around 15% of relative contrast variation. If the change is an increase /decrease then an ON/OFF event is generated by the pixel (see **Figure 1**). This asynchronous way of coding allows to convey the timing of the events with a

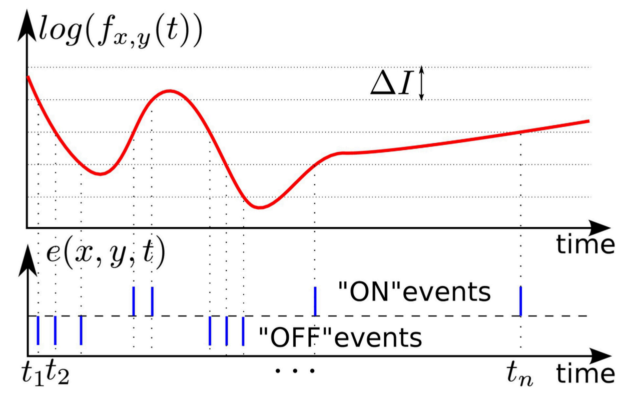


FIGURE 1 | Illustration of event-based encoding of visual signals. Shown are the log-luminance measured by a pixel located at $(x, y)^T$ and the asynchronous temporal contrast events signal generated by the DVS with respect to the predefined threshold ΔI.

high temporal resolution ($\sim 1\ \mu$s). The "effective frame rate" of such pixels is several kHz. We define an event occurring at time t at the pixel $(x, y)^T$ as:

$$e(x, y, t) = |p| = 1, \tag{1}$$

where p is the polarity of the event. p equals 1 ("ON") whenever the event signals an intensity increase, or -1 ("OFF") for a decrease, but for the purposes of this article the polarity is not used. This data-driven representation reduces redundancy in the visual input, and maintains the encoding of exact times of input signal changes, which allows very high temporal dynamics of acquisition.

2.2. GENERAL ARCHITECTURE

Figure 2 shows the general architecture of the feature selection process. In the following we briefly describe the overall architecture, with more detailed descriptions of the individual components below. To capture the temporal dynamics of spatiotemporal features, we use Echo-State Networks (ESN) (Jaeger, 2002) that act as predictors of future outputs. To achieve unsupervised learning of distinct features we use multiple ESNs that compete for learning and detection via a WTA network. As the first stage, the signal coming from the DVS retina is preprocessed, by converting the DVS output into analog signals as required by the ESNs' structure. In the second stage, labeled *ESN layer* in **Figure 2**, each ESN receives the converted output of the DVS to predict its evolution one timesteamp in the future. The readout of each ESN is trained for this task, and each network should learn to predict different temporal dynamics. To achieve this, the next layer of the architecture, labeled *WTA with Predictability minimization* in **Figure 2**, implements a Winner-Take-All (WTA) neural network, which selects the best predictor from the available set of predicting ESNs. Through competition, the WTA inhibits poorly predicting ESNs to ensure that the best predictor has sufficient time to learn a particular spatiotemporal sequence. This layer also contains a predictability minimization process to promote orthogonality of predictions between the different ESNs. The selected ESN is then trained to recognize the spatiotemporal pattern, and learns to predict its temporal evolution. The WTA competition ensures that each ESN specializes on an independent feature, thus preventing two ESNs from

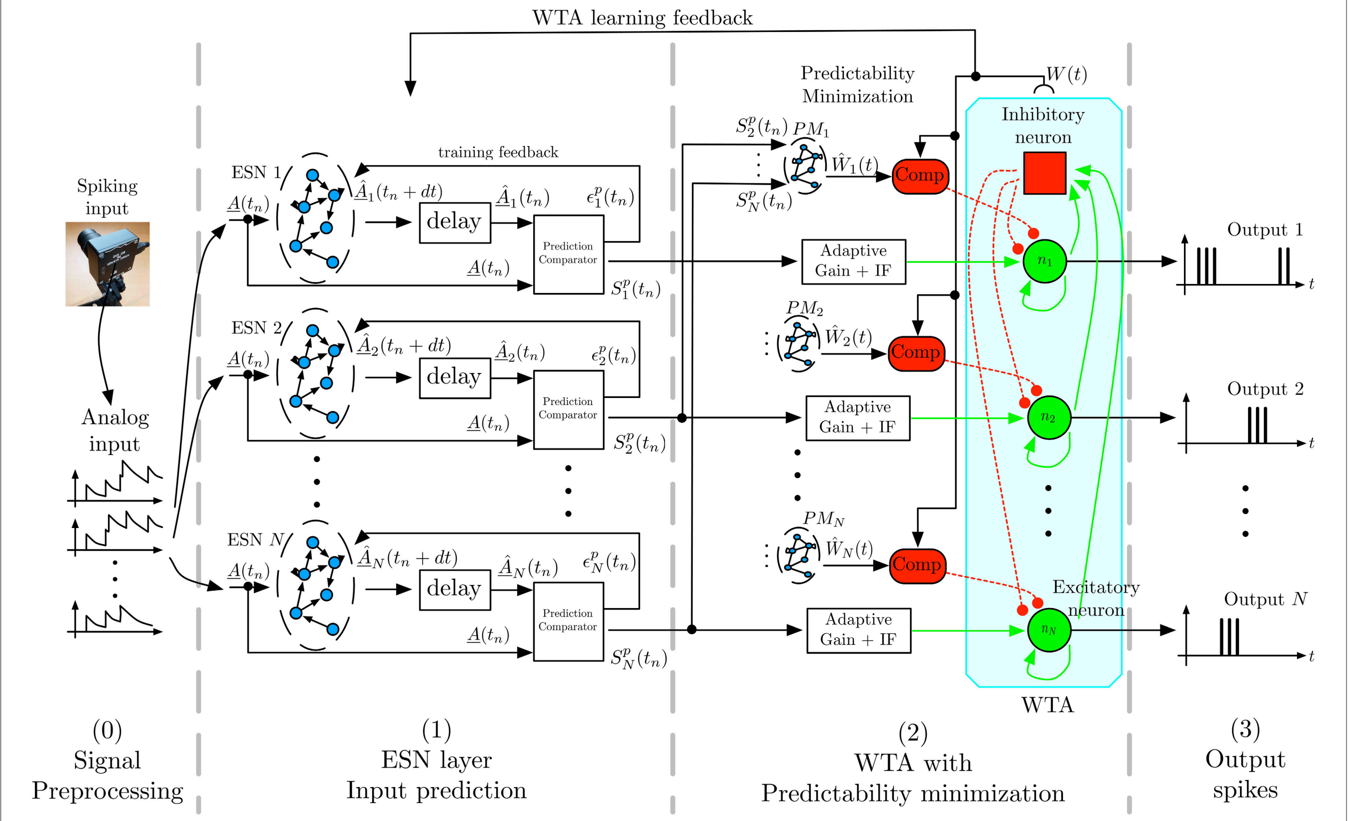


FIGURE 2 | Architecture for unsupervised spatiotemporal feature extraction. Spikes from the DVS are transformed by filtering into analog input signals that are sent to a set of ESN networks. Each ESN is trained to predict future input activations based on current and past activities. The prediction is compared to the actual inputs, and the output signal S_k^p, which is a representation of the ESN's prediction performance is fed into a Winner-Take-All (WTA) network. This WTA selects the best predicting ESN and enables it to train on the present input sequence. A predictability minimization process promotes orthogonality of predictions between the different ESNs during the WTA selection. The combination of temporal prediction and competition through the WTA allows each ESN to specialize on the prediction of a distinct dynamical feature, which thus leads to learning of a set of different feature detectors.

predicting the same pattern. Consequently, at any given time, the winning network in the WTA layer indicates the detected feature. Through random initialization of ESNs and WTA competition, the architecture extracts distinct spatiotemporal features from event-based input signals in a completely unsupervised manner.

For the experiments described in this article, the architecture has been fully implemented in software, using DVS recordings of real-world stimuli as inputs. In particular, the visual inputs for all experiments contain the typical noise for this kind of sensor, and do not use idealized simulated data.

2.3. SIGNAL PRE-PROCESSING

The DVS retina has approximately $16K$ pixels in total. Directly using each pixel as an input to the ESN reservoir would require a network with $16K$ input neurons, and, in typical reservoir computing setups, 10–100 times more hidden neurons. Since this is a prohibitively large size for real-time simulation of neural networks on conventional current computers, a pre-processing stage is introduced to downsample the dimensionality of the input. Please note that this is not a fundamental requirement, since especially future large-scale neuromorphic processors and other dedicated hardware platform could potentially handle real-time execution of such large networks (see Discussion), but this is beyond the scope of our proof-of-principle study.

Figure 3 provides a more detailed view of the first layer of the architecture, named layer (0) in **Figure 2**. To reduce the input dimensionality of the DVS signal, the retina pixels are first spatially resampled into cells $C(x_c, y_c)$ of $\delta_x \times \delta_y$ pixels, each integrating pixels around the center (x_c, y_c) according to:

$$C(x_c, y_c) = \left\{ (x, y) \quad \middle| \quad \begin{matrix} x \in [x_c - \delta_x, x_c + \delta_x] \\ y \in [y_c - \delta_y, y_c + \delta_y] \end{matrix} \right\}. \qquad (2)$$

Next, the signals are quantized by introducing spatiotemporal receptive fields $RF(x_0, y_0, t_1, t_2)$, covering $\Delta_x \times \Delta_y$ subsampling cells, which collect all events in a spatiotemporal volume in the time interval $[t_1, t_2]$ according to:

$$RF(x_0, y_0, t_1, t_2) = \qquad (3)$$
$$\left\{ e(x, y, t) | t \in [t_1, t_2], (x, y) \in C(x_c, y_c), \begin{matrix} |x_c - x_0| \leq \Delta_x \\ |y_c - y_0| \leq \Delta_y \end{matrix} \right\}.$$

Conversion of events into analog signals is achieved by filtering with a causal exponential filter with time constant τ, defined as $G(t, t_i) = e^{-(t - t_i)/\tau} \cdot H(t - t_i)$, where $H(t)$ is the Heaviside function, which is 1 for $t \geq 0$ and zero otherwise. This filter is applied to all spikes coming from pixels (x, y) contained in a receptive field $RF(x_0, y_0, t_0, t)$, yielding the analog output signal A, which is fed into the ESNs:

$$A(x, y, t_0, t) = \sum_{\substack{e(x_i, y_i, t_i) \in RF(x_0, y_0, t_0, t) \\ x_i = x, \; y_i = y}} G(t, t_i) \quad , \qquad (4)$$

where t_0 is a chosen time origin.

The complete preprocessed input at time t fed into the ESN layer is the vector formed by all outputs $A(x, y, t_0, t)$ of pixels contained in $RF(x_0, y_0, t_0, t)$. For clarity, we will in the following only consider a single receptive field denoted as $\underline{A}(t)$:

$$\underline{A}(t) = \begin{pmatrix} A(x_1, y_1, t_0, t) \\ \vdots \\ A(x_M, y_M, t_0, t) \end{pmatrix} \qquad (5)$$

2.4. ESN LAYER—INPUT PREDICTION

This layer (**Figure 2**-(1)) computes the prediction of input signals for N different ESNs (Jaeger, 2002). The k^{th} ESN is defined by its internal state s^k, and the three weight matrices W^k_{out} (for output or *readout* weights), W^k_{in} (for input weights), W^k_{back} (for feedback weights), and the recurrent weights W^k_{r}. These weight matrices are initialized randomly for each ESN and encoded as 64 bit floating-point numbers. The internal state s^k of the ESN and its output (out^k) are iteratively updated, and evolve according to :

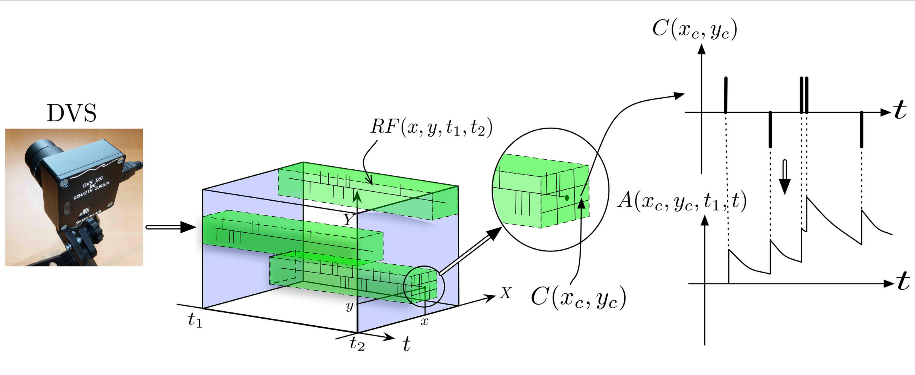


FIGURE 3 | Illustration of signal pre-processing to convert DVS events into equivalent analog input for ESNs. In order to reduce the number of input channels to the system and reduce computational load, the input from retina pixels is first spatially subsampled into cells $C(x_c, y_c)$. Each set of ESNs then receives input from a particular receptive field $RF(x, y, t_1, t_2)$. To compute the equivalent analog input, an exponential kernel finally is applied to each event contained in the receptive field.

$$s^k(t_n) = f\left(W_r^k \cdot s^k(t_{n-1}) + W_{in}^k \cdot \underline{A}(t_n) + W_{back}^k \cdot out^k(t_{n-1})\right), \tag{6}$$

$$out^k(t_n) = f^{out}\left(W_{out}^k \cdot s^k(t_n)\right). \tag{7}$$

In our experiments, the logistic function is used as the non-linearity f for the internal state evolution, and a linear readout is used as f^{out}. Every ESN is trained to predict its future input at one timestep ahead (i.e., at $t_n + dt$), thus the output of the ESN according to Equation 7 creates a prediction $\hat{\underline{A}}_k(t_n + dt) = out^k(t_n)$, which should match $\underline{A}(t_n + dt)$. As is usual for ESNs, only the readout weights W_{out}^k are adapted, the recurrent and other weights are kept at their random initial values which are drawn from uniform distributions.

As suggested in Jaeger and Haas (2004), training of the readout weights W_{out}^k can be achieved with a standard recursive least squares algorithm (here a version described in Farhang-Boroujeny (2013) was used). This algorithm recursively adapts W_{out}^k so as to minimize a weighted linear least squares cost function, computed from the prediction error:

$$\epsilon_k^p(t_n) = |\hat{\underline{A}}_k(t_n) - \underline{A}(t_n)|. \tag{8}$$

This method is well-suited for online learning, since the coefficients of W_{out}^k can be updated as soon as new data arrives.

The output of the ESN layer into the subsequent WTA layer is a similarity measure $S_k^p(t_n)$ for each ESN, which indicates the quality of each prediction for the currently observed input:

$$S_k^p(t_n) = \frac{\sum_i \left|\underline{A}(t_n)_i \cdot \hat{\underline{A}}_k(t_n)_i\right|}{\sum_i \left|\underline{A}(t_n)_i\right| \cdot \sum_i \left|\hat{\underline{A}}(t_n)_i\right|}, \tag{9}$$

where i is summing over all components of $\underline{A}(t_n)$ and $\hat{\underline{A}}(t_n)$, which have been properly normalized to take on values between 0 and 1.

2.5. WINNER-TAKE-ALL SELECTION

Based on the indicators of prediction quality $S_k^p(t_n)$ computed by the ESN layer, the third layer of the model (**Figure 2**-(2)) selects the best predictor among the N ESNs through a WTA mechanism. The WTA network consists of a set of N neurons $\{n_1, \ldots, n_N\}$ plus an inhibitory neuron, which is recurrently and bi-directionally connected with the excitatory neurons, as detailed in Coultrip et al. (1992), Douglas et al. (1994), Liu and Oster (2006), and Oster et al. (2009). The task of the WTA is to select from the pool of ESNs the one whose prediction best matches the actual dynamics of the present input, and which thus has the highest similarity $S_k^p(t_n)$, as computed by layer (1) in **Figure 2**.

Inputs to the WTA neurons are generated from the S_k^p values using non-leaky Integrate-and-Fire (IF) neurons, which transform the analog values into spike trains. To make the WTA network more robust to the variations in the similarity measure, a sigmoid function is applied to the S_k^p values to compute the input current fed to the IF neurons:

$$g_{IF}(S_k^p) = G_{min} + \frac{G_{max} - G_{min}}{1 + \exp(-(S_k^p - x_0)/\lambda)}. \tag{10}$$

G_{min} and G_{max} define the interval in which the output firing rates of the IF neurons are taking values. They are set experimentally to achieve spike rates spanning from 5 kHz to 15 kHz. λ sets the selectivity of the sigmoid which is an increasing function of λ (λ has been experimentally tuned to $5.0e^{-5}$ in our experiments). The value of the offset x_0, which is subtracted from the S_k^p is managed by a proportional controller. Its input reference is set such that x_0 approaches the value of S_k^p output by the selected best predictor. This ensures that whatever the current state of the system is, the sigmoid g_{IF} is always centered on the current value of interest, giving the best selectivity possible to detect changes in the best predictor. The update period of this controller is set to 0.5 ms. The index of the spiking neuron from the WTA network then corresponds to the best predictor $W(t)$ satisfying :

$$W(t) = \underset{k \in \{1,\ldots,N\}}{\operatorname{argmax}} \; g_{IF}(S_k^p(t)) \quad . \tag{11}$$

The obtained index $W(t)$ is used to drive the learning process of the *ESN layer*. Only the ESN selected by the WTA network (ESN with index $W(t)$) is trained on the input signal. This adaptive WTA achieves good performance in the selection of the best predictor even if the similarity measurement has a large variance (this happens for instance if the system is exposed to a set of very different stimuli).

This setup of the WTA architecture always generates outputs, even if no input is present. This potential inefficiency can be avoided by adding another output layer, which computes a gating function that depends on the global input activity. Using this mechanism, output neurons driven by the output of the WTA will only fire if in addition the input activity is bigger than a defined threshold. The threshold can be either defined on the average event rate, or the average value of $\underline{A}(t_n)$.

2.6. PREDICTABILITY MINIMIZATION

The third layer implements, in addition to the WTA selection, a predictability minimization algorithm, which ensures that each ESN specializes in predicting different features in the input. It implements a criterion suggested by Barlow (1989) and Schmidhuber (1991) to evaluate the relevance of the prediction of each ESN: an ESN's prediction is considered relevant if it is not redundant given the other ESNs' predictions. This predictability minimization step promotes orthogonality of predictions between the individual ESNs, and encourages a maximally sparse representation of the learned input classes, thereby achieving good coverage of the presented input space. For each ESN k, an estimator $\hat{W}_k$ of the WTA output is used, which receives only the similarity measures $S_{k'}^p$ of the other ESNs as input. For a consistent framework of estimators and predictors, we chose to use ESNs (named $PM_1, \ldots, PM_N$ in **Figure 2**) to implement the $\hat{W}_k$ estimator. This also allows taking into account the highly dynamic information contained in the input data recorded with the DVS. Training of the ESNs follows the same principles as described in Section 2.4.

If the estimator $\hat{W}_k$ and the WTA output agree, i.e., $\hat{W}_k(t_n) = W(t_n)$, then this means that the kth ESN is not currently learning a new feature, because the same information can also be deduced from the output of the other ESNs. In this case, the corresponding neuron of the WTA is inhibited to prevent this ESN from learning the currently presented input patter. The inhibition also causes the output of the WTA to stop responding to the input, thus promoting another one.

3. RESULTS

3.1. EXPERIMENTAL SETUP

The experiments presented in this article were performed with the setup shown in **Figure 4A**. It consists of a DVS retina observing a treadmill, on which moving bars with 9 different orientations move across the field of view of the DVS at constant speed. For the experiments, the recurrent connectivity matrix W_r for each ESN was initialized randomly, and rescaled to have spectral radius 0.7, which fulfills the Echo State Property (Jaeger, 2002). The other weight matrices were randomly chosen from a uniform distribution in $[-0.4; 0.4]$ for W_{in}, $[-0.02, 0.02]$ for W_{back} and $[-0.01, 0.01]$ for W_{out}. The pre-processing uses exponential kernels with a time constant of 10 ms.

3.2. SINGLE RECEPTIVE FIELD

The first experiment uses 8 ESNs, each composed of 15 analog neurons, randomly connected in the reservoir. Only one RF, consisting of 17×17 cells $C(x_c, y_c)$ spanning 5×5 pixel is used as input to each ESN. **Figure 4B** shows the different predictions of the ESNs in response to an input signal. The WTA succeeds in selecting the best predicting network for the current input. **Figure 4C** shows for each stimulus the best predictions and the associated ESN. As expected, the results confirm that every network has specialized in the prediction of the temporal evolution of a specific oriented moving pattern. Since natural scenes contain many independent features, which are likely to occur in larger numbers than the number of available ESNs, we tested here the performance of an architecture with only 8 ESNs for 9 different patterns of moving oriented bars. The results indicate that some of the ESNs tend to learn more than one dynamic feature, so that the system can represent all input features as accurately as possible. In order to select the most appropriate number of predictors, additional control mechanisms could be employed. An example of this is the response of ESN1, which is the best predictor both for pattern 8 and 9 (**Figure 4C**).

Figure 5 shows the output of the same system for three successive testing presentations of the stimulus. We can see that each ESN is responding to a specific orientation of the bars. Moreover, the process is repeatable over the three presentations with a difference in the temporal span of the responses. This is due to the increase of the translation speed of the bars during the recording to show that the networks effectively respond to the bar's orientations independently of their speed.

Figure 6 shows the prediction error of each ESN during several presentations of the stimulus. The output of the WTA network is shown below each curve, indicating when a particular ESN is selected as the best predictor. An ESN is correctly selected whenever its prediction error is the lowest. Periods in which all prediction errors are close to zero correspond to periods without input (shown as gray regions in the figure). This is a result of the approximate linearity of the ESNs and their low spectral radius: when only weak input is fed into the network, the ESNs readout output also approaches zero, which results in a low prediction error for times when no stimulus is presented (the only input to the networks then is background noise from the DVS pixels).

3.3. MULTIPLE RECEPTIVE FIELDS

In the second experiment, the field of view of the DVS is split into 3×3 smaller RFs of identical size (9×9 cells of 3×3 pixels), as

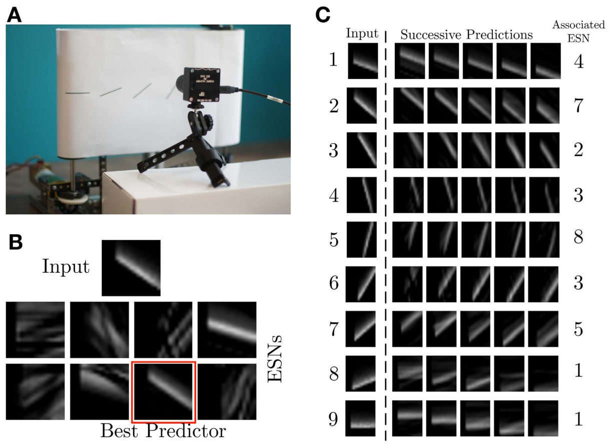


FIGURE 4 | Experimental recording setup. (A) A DVS records patterns moving on a treadmill. **(B)** Current input pattern (top) and predictions of the different ESNs at a given time. The best predictor is highlighted in red. **(C)** Results of ESN training. The left column shows a snapshot from each of the nine different patterns. The plots to the right show different predictions for different time steps in the future, obtained from the ESN which is specialized in the given pattern. The time difference between the five predicted patterns is 0.01 s.

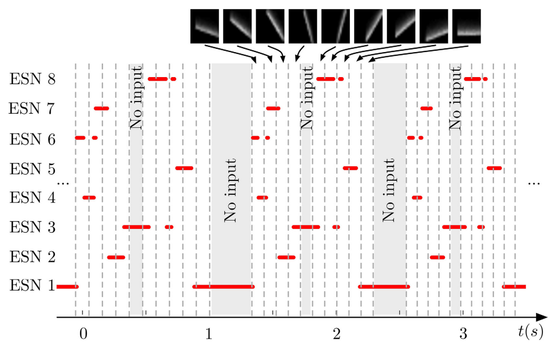


FIGURE 5 | Output of the WTA network during repeated presentation of a series of nine input patterns. Red lines indicate when an ESN was selected by the WTA network. Dashed vertical lines mark time points when the input presented to the DVS changed from one pattern to another (the 9 patterns are shown on top of the figure). Shaded areas indicate times when no stimulation was present. Every ESN learns to respond to only a small subset of input patterns (typically exactly one pattern). This response is reproducible over different stimulus presentations.

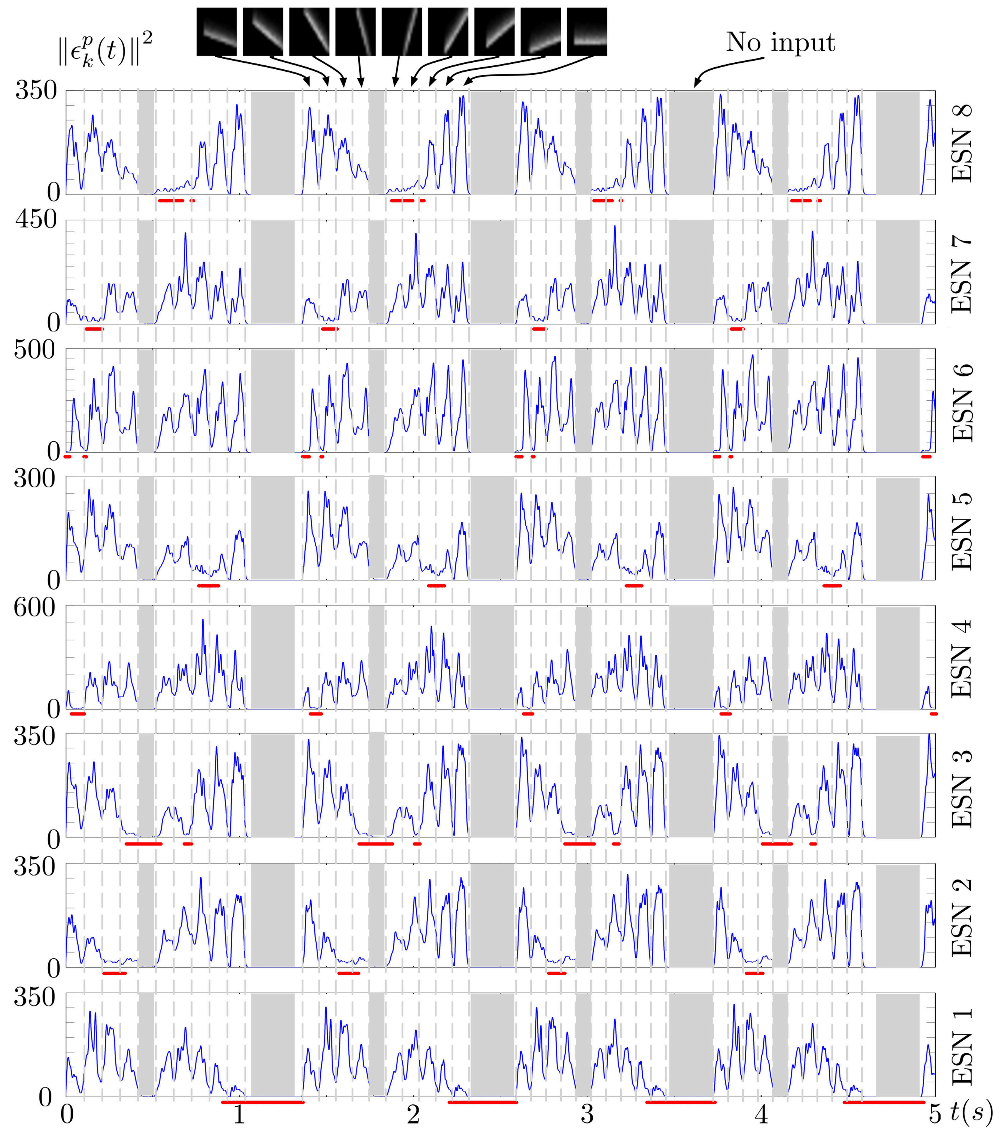


FIGURE 6 | Prediction error of the 8 ESNs during several presentations of the input stimulus. Red lines below each plot shows the output of the WTA neuron corresponding to each ESN, thus indicating times when each ESN was selected as the best predictor. We can observe that ESNs are correctly selected when their prediction error is minimal amongst all the networks.

shown in **Figure 7**. This shows the full intended behavior of the system as a local spatiotemporal feature detector, in which different features can be assigned to small receptive fields covering the entire field of view of the sensor (instead of being covered by only one big one RF like in **Figure 4**). For each RF 8 ESNs are used as feature detectors. In the learning phase, they are trained only with the input to the central RF. Subsequently, their weights are copied and the ESNs are used independently for all 9 RFs. Thus, all RFs have ESNs with identical weights (and so detects the same features), but receive different inputs and therefore evolve independently. **Figure 7A** shows different snapshots of the DVS recording for an oriented bar moving across the field of view. The output of the predictors for each RF is shown in **Figure 7B**, while **Figure 7C** indicates for each RF the index of the ESN selected. The figure also shows that ESN predictors are only selected when there is substantial input activity in the RF. As in the previous experiment, dynamic feature selection is reproducible and exhibits precise timing, as shown in **Figure 8**. Here, only the 3 RFs on the middle line of the input space are shown. Because the input stimulus moves horizontally, the outputs of the WTA circuits are similar, with a little time delay. Using multiple smaller RFs instead of one is also a potential solution to represent more features with a finite set of ESN. The feature descriptor is then a combination of the outputs of all available ESNs, which need to be processed by another layer. This is however, beyond the scope of the present paper.

Choosing the right number of ESNs for the feature detection architecture is not always straightforward, and depends on the

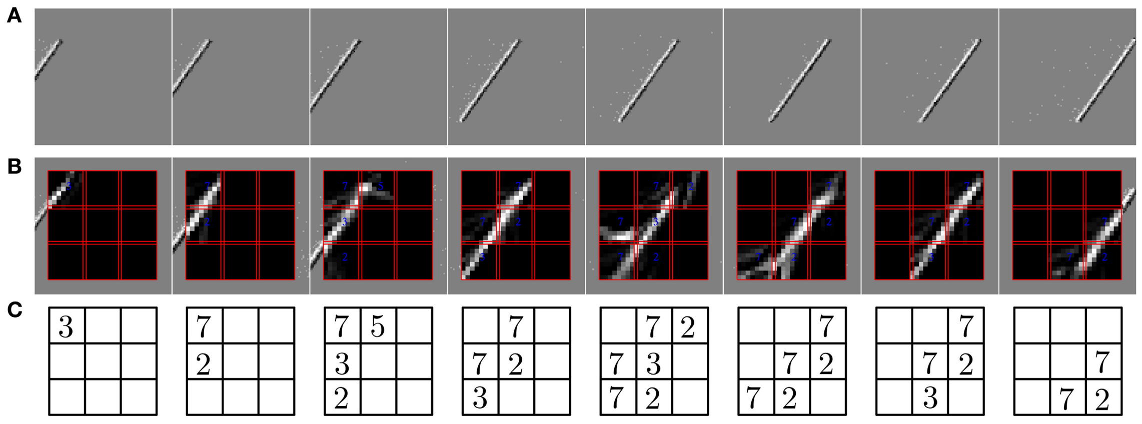


FIGURE 7 | Predictions of the ESNs when the camera's field of view is divided into multiple receptive fields (RFs). (A) Output of the DVS for a bar moving across the field of view. **(B)** Predictions of ESNs processing different RFs (indicated by red boxes). **(C)** Index of the best predicting ESN in every RF at the timepoints of the snapshots. Indices are only shown if the input activity in the RF exceeds a given threshold.

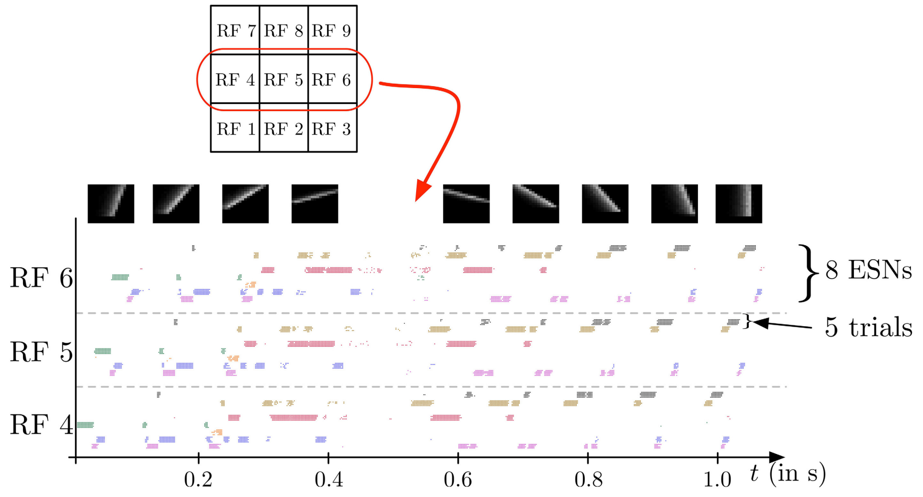


FIGURE 8 | Spike output for the WTA neurons corresponding to the eight ESNs for each of the 3 central RFs (see Figure 7) during the presentation of a series of 9 moving bars with different orientations (see snapshots on top). Dots indicate the times of output spikes for 5 repeated stimulus presentations, which are drawn on different coordinates along the y-axis, but grouped by WTA neuron. Each ESN, depending on its index in the RF, is associated with a color used to represent the dots corresponding to its output. The results show a highly reproducible response of the feature detectors for different trials, and also similar time delays for different stimulus presentations. Because the input stimulus moves horizontally, the outputs of the WTA circuits are similar, with a little time delay. Note that only one ESN can be active at any given time in each RF. Apparent simultaneous spikes from multiple WTA neurons are due to the scaling of time in the horizontal axis of the figure.

number of distinct features present in a scene. In **Figure 5** it was shown that when the number of ESNs is smaller than the number of features, an ESN can learn multiple features instead of one. **Figure 9A** shows the number of steps in which each ESN is trained if 8 ESNs are trained on 9 different input patterns. It is shown that all networks are trained for a similar number of epochs. When instead the number of ESNs exceeds the number of features, we find that only the minimum necessary number of predictors is selected, and the remaining ESNs are still available to learn new features, should there be distinct future visual inputs. **Figure 9B** a clear specialization of ESNs, if 20 networks are used to encode the same 9 features that were used in **Figure 9A**. Only 9 out the 20 ESNs show increased activation during the stimulation presentations.

3.4. COMPLEX INPUT STIMULUS

In the last experiment, the ability of the architecture to represent more complex features was tested. Instead of using oriented bars, we now present digits (from 1 to 9) to the camera, with a single receptive field covering the whole stimulus. Nine ESNs were used in the system, which matches the number of distinct patterns. To make them visible for DVS recordings, the nine digits were animated, by hand, with a random jittering movement around a central spatial position. This was intended to

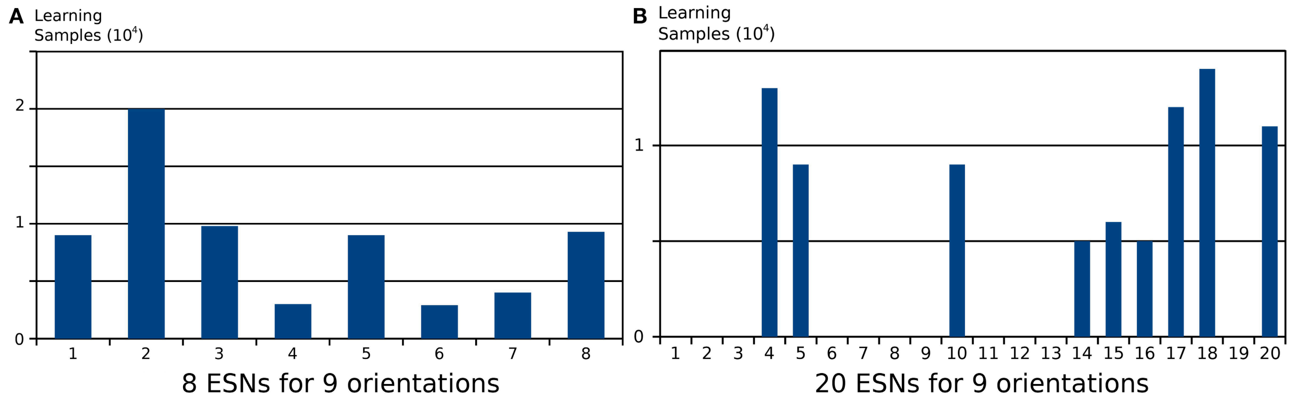


FIGURE 9 | Number of learning samples per reservoir for two different architectures applied to the same input. Every step where training of the ESN readout was activated is counted as a learning sample. **(A)** Learning samples for 8 ESNs trained on 9 different input patterns. **(B)** Learning samples for 20 ESNs trained on the same 9 input patterns. The results show that when the pool of ESNs is bigger that the number of features present in the input, only a necessary subset of ESNs from the pool is used to learn these features. The remaining ESNs are not trained, and can be used to learn new features from future inputs.

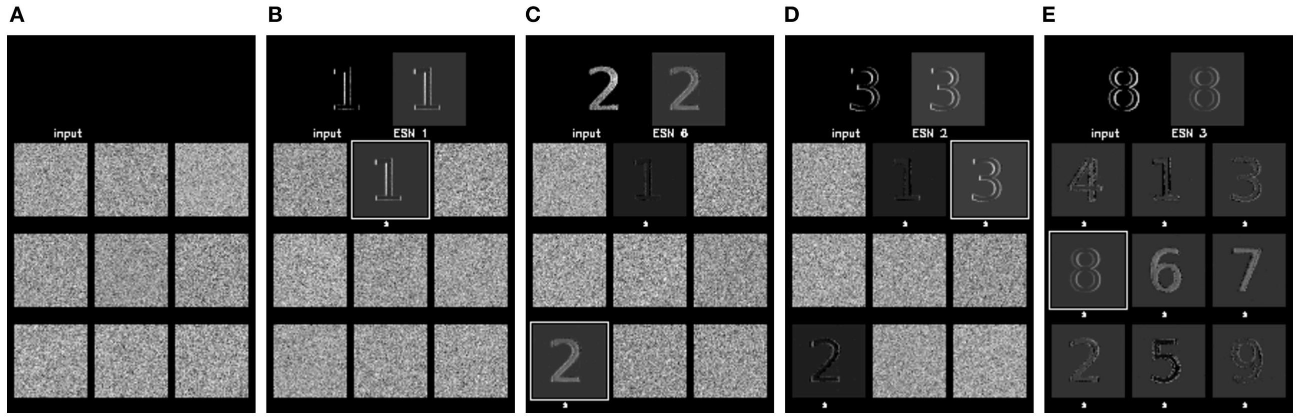


FIGURE 10 | Learning of more complex feature detectors, showing the output of the system when presented with stimuli composed of digits from 1 to 9. Each snapshot on the top left shows the analog input to the ESN, obtained by filtering the DVS events. The top right shows the prediction of the best ESN, as selected by the WTA circuit. Below, the current predictions of all nine ESNs used in the experiment are shown. A white square around the prediction indicates the ESN that was selected by the WTA. The snapshots show the progression of learning, starting with an untrained network in **(A)**, which only produces random predictions. **(B–D)** show the output of the same networks after the presentation of patterns "1," "2," and "3," (respectively). A white mark underneath an ESN prediction indicates that this ESN has learned a feature. Finally **(E)** shows its output after the end of the learning process where all networks have learnt an input stimulus.

simulate what would be seen by the retina when the eye follows microsaccadic movements. Because the jitter is random, the input stimulus mainly contains spatial information. This experiment allows us to test the robustness of the proposed method to several spatiotemporal patterns, including the degenerate case where only one spatial information is relevant for the feature. Some snapshots of the system's output are shown in **Figure 10**.

In the first stage of the experiment, the system is presented with visual stimuli of the digits 1–9, in this order. The images at the top of the plot shows the input to the receptive field at the time of the vertical dotted lines. Each number is presented for 5 s, followed by a pause of 3 s, in which no input is presented. In **Figure 11** the learning phase is marked by a gray shaded background. Next, two test sequences are presented to the DVS: The *Test 1* sequence is composed of the random sequence "1 3 5 7 9 2 4 6 8," using the same presentation and pause times as in the learning phase. The *Test 2* sequence is composed of another random sequence "9 8 7 6 5 4 3 2 1," this time without pauses between digit presentations (which still last for 5 s). These sequences are represented as the ground truth for the experiment by blue horizontal lines in **Figure 11**. For clarity, we re-ordered the ESNs such that the ESN index corresponds to the digit it represents. Successful learning means that the blue lines should align as much as possible with the red dots, representing the output of the WTA network. Occasional deviations are due to noise.

Figure 11 shows that each ESN manages to learn complex features, and reliably recognizes them when the respective feature is presented again. This was achieved with raw, noisy DVS inputs,

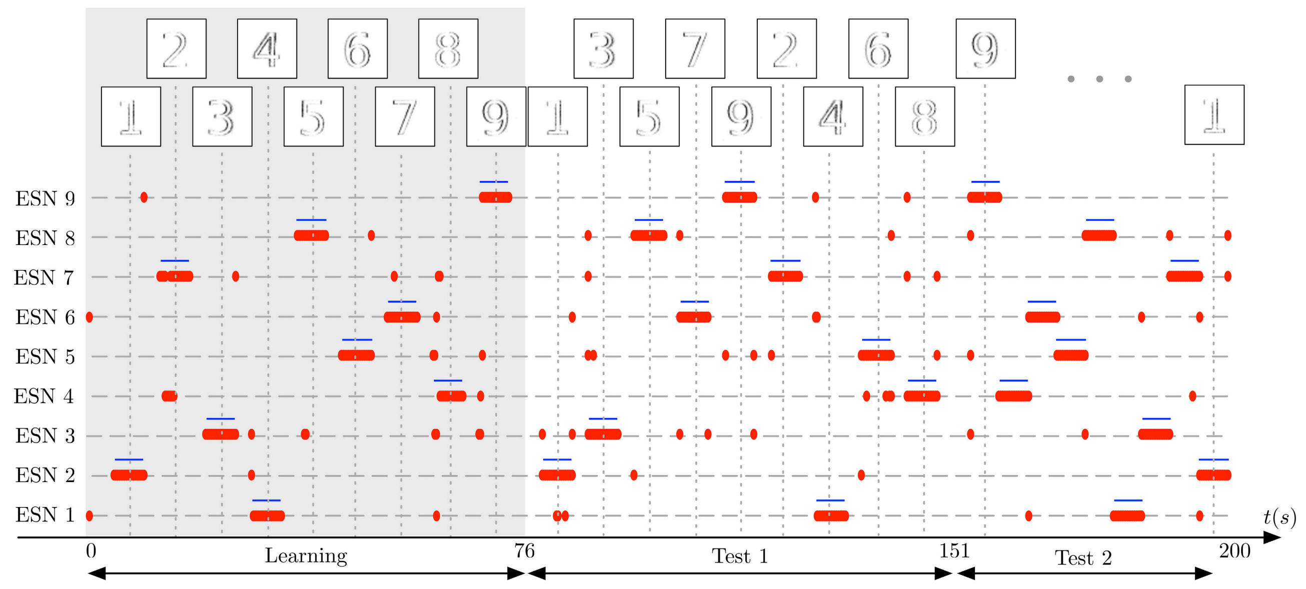


FIGURE 11 | Learning complex features from DVS inputs. The DVS records digits from 1 to 9, each animated with random jitter simulating the effect of microsaccades in the biological eye. Blue horizontal lines show the ground truth, indicating when each number was present in the input, and thus when a specific ESN should fire. The corresponding pattern presented to the camera at that time is represented on top of the figure. Red dots show time points where each ESN is selected by the WTA network. The gray shaded area marks the learning phase, in which every digit from 1 to 9 was presented once to the system. Subsequently, two series of tests are shown: First, in the period marked as *Test 1*, the 9 digits are presented in random order, but with short pauses between stimulus presentations. Then, in the period marked as *Test 2*, the digits are presented again in a different random order, but without pauses in between. The results show that also for complex features like digits, every ESN can learn to specialize and represent a distinct feature.

and fully random jitter of the digits during presentation. The experiment shows that complex features can be extracted and recognized also in the absence of characteristic spatiotemporal structure in input patterns.

4. DISCUSSION

This article presents a new architecture for extracting spatiotemporal visual features from the signal of an asynchronous event-based silicon retina. The spatiotemporal signal feeds into the system through a layer of ESN, which compute predictions of future inputs. An unsupervised learning process leads to specialization of ESNs to different features via WTA competition, which selects only the best predictors of the present input pattern for training. Whenever an already learned pattern is presented again, the system can efficiently and reliably detect it. Experimental results confirm the suitability of the feature extraction method for a variety of input patterns. The spatiotemporal feature extraction leads to robust and reproducible detection, which is a key requirement for its use in higher-level visual recognition and classification. A central characteristic of the presented technique, in contrast to conventional computer vision methods, is that it does not depend on the concept of representing visual inputs as whole image frames. Instead, the method works efficiently on event-based sparse and asynchronous input streams, which maintain the temporal dynamics of the scene due to the highly precise asynchronous time sampling ability of the silicon retina. Thus, also the extracted spatiotemporal features contain richer dynamic information, in addition to recognizing spatial characteristics.

Central to the definition of spatiotemporal features in our architecture is the presence of multiple models for prediction, which compete already during learning, such that specialization can occur. Similar concepts are used by various well-known machine learning frameworks, most notably the mixture-of-experts architecture (Jacobs et al., 1991; Jordan and Jacobs, 1994; Yuksel et al., 2012), in which a gating function creates a soft division of the input space for multiple local "expert" models. The output of the whole network is then a combination of the expert predictions, weighted according to their responsibility for the present input. These architectures have been extended in brain-inspired architectures for reinforcement learning and control (Haruno et al., 2001; Doya et al., 2002; Uchibe and Doya, 2004), where multiple forward models and controllers are learned simultaneously, and the prediction performance of the forward model determines the selection of the most appropriate local controller. Mixture-of-experts architectures are closely related to learning mixture models with the EM algorithm (Dempster et al., 1977; Jordan and Jacobs, 1994), where the E-step computes a soft assignment of data points to models. Nessler et al. (2009) and Nessler et al. (2013) have proven that this can be implemented in spiking neural networks, using a soft WTA circuit to compute the E-step, and an STDP learning rule to implement the M-step. Compared to these related architectures, our new model advances in three important aspects: Firstly, whereas EM and mixture-of-experts address static input distributions, we here extend this to multiple feature predictors for spatiotemporal sequences. Secondly, our architecture allows online learning of independent features, which contrasts with batch methods like PCA or ICA that operate on the full dataset after its collection. Thirdly, our neural network architecture is specifically designed to work with spiking inputs and for implementation with spiking neurons, thus maintaining the precise dynamics of event-based vision sensors. Other spiking neural network architectures for processing DVS inputs such as spiking ConvNets (Farabet et al., 2012; Camuñas-Mesa et al., 2014), and spiking Deep-belief networks (O'Connor et al., 2013) do not explicitly model the dynamics of the features extracted within the networks, but instead rely on different conversion mechanisms from analog to spiking neural networks, without taking sensor dynamics into account. The features they extract are thus characterizing a current snapshot of the input, and do not take its future trajectory into account like the ESN predictors of the presented model, but nevertheless are very useful for fast recognition. This is also true for approaches that directly classify spatiotemporal spike patterns, see e.g., (Sheik et al., 2013; Tapson et al., 2013). Spiking network models that represent spatiotemporal dynamics by emulating Hidden Markov Models have recently been introduced (Corneil et al., 2014; Kappel et al., 2014). Compared to our approach, these networks do not directly learn dynamic input features, but rather identify hidden states to determine the position within longer sequences.

The combination of visual sensing with bio-inspired artificial retinas and event-based visual feature extraction, as presented in this article, opens new perspectives for apprehending the mechanisms of visual information encoding in the brain. It is clear that the traditional views of visually selective neurons as static image filters for receptive fields, e.g., as Gabor-like orientation filters, which are central to many classical vision models like HMAX or Neocognitron (Fukushima, 1980; Serre et al., 2002), fails to explain how these neurons deal with the highly dynamic and sparse spike inputs from biological retinas. In the presented approach, features are naturally learned and adapted to the task. In **Figure 9** it was shown that if the number of available ESNs exceeds the number of features necessary to describe a scene, only the minimum necessary number of networks are trained. This has the desirable effect that whenever a new scene with new features is encountered, the previously unused ESNs can be trained to predict novel stimulus features. This behavior has several benefits: firstly, the number of ESNs does not have to be precisely tuned, but can be set to the highest acceptable number, and only the minimum number of networks is actually recruited and trained as feature detectors by the system. Alternatively, one could employ a different strategy in which new networks are recruited to the pool, whenever all current ESNs have specialized on features. Secondly, training of feature detectors works completely unsupervised, so no higher-level controller is needed to identify what the elementary features for a scene should be. Although the precesence of a supervisor is not necessary, having such information available would still be beneficial. For instance, another processing layer could use the outputs of the WTA to control the survival of each network. If such processing layer determines that a particular network does not provide enough interesting information, the supervisor could decide to reset and release the associated ESN, so that it can detect more relevant features.

The presented method has great potential for use in event-based vision applications, such as fluid and high-speed recognition of objects and sequences, e.g., in object and gesture recognition (O'Connor et al., 2013; Lee et al., 2014), or for high-speed robotics (Conradt et al., 2009; Mueggler et al., 2014).

The presented architecture is almost entirely based on computation with spikes. Inputs come in the form of AER events from DVS silicon retinas, providing an event-based representation of the visual scene. The WTA circuit for choosing between feature extractors is also working with spikes, and produces spike outputs, which indicate the identity of the detected feature. The only component of the system which does not entirely use spikes is the layer of ESNs that predict the visual input, but this restriction could be lifted by replacing ESNs with their spiking counterparts, called Liquid State Machines (LSMs) (Maass et al., 2002), which are computationally at least equivalent to ESNs (Maass and Markram, 2004; Büsing et al., 2010). The reasons why we have chosen to use ESNs for this proof-of-principle study are the added difficulty of tuning LSMs, due to the larger number of free parameters for spiking neuron models, delays, or time constants, in addition to the higher computational complexity involved in the simulation of spiking neural networks on conventional machines, which makes it hard to simulate multiple LSMs in real-time. Overall, we expect the improvement due to using fully spike-based feature detectors and predictors to be rather minor, since the ESNs can be efficiently simulated at time steps of 1 ms, which is also the time interval at which the silicon retina is sending events through the USB bus. However, a fully spike-based architecture does have great advantages in terms of efficiency and real-time executing if it can be implemented entirely on configurable neuromorphic platforms with online learning capabilities (Indiveri et al., 2006; Galluppi et al., 2014; Rahimi Azghadi et al., 2014), which is the topic of ongoing research.

ACKNOWLEDGMENTS

This work was performed in the frame of the LABEX LIFESENSES [ANR-10-LABX-65] and was supported by French state funds managed by the ANR within the Investissements d'Avenir programme [ANR-11-IDEX-0004-02]. XL has been supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n ° 604102 (HBP). MP has been supported by the Swiss National Science Foundation Grant 200021_146608 and the Samsung Advanced Institute of Technology. The work proposed in this paper is resulted from discussions at the Capocaccia and Telluride Workshops; the authors would like to thank the sponsors and the organizers.

REFERENCES

Barlow, H. B. (1989). Unsupervised learning. *Neural Comput.* 1, 295–311. doi: 10.1162/neco.1989.1.3.295

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* 110, 346–359. doi: 10.1016/j.cviu.2007.09.014

Berry, M., Warland, D. K., and Meister, M. (1997). The structure and precision of retinal spike trains. *Proc. Natl. Acad. Sci. U.S.A.* 94, 5411–5416. doi: 10.1073/pnas.94.10.5411

Büsing, L., Schrauwen, B., and Legenstein, R. (2010). Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural Comput.* 22, 1272–1311. doi: 10.1162/neco.2009.01-09-947

Camuñas-Mesa, L. A., Serrano-Gotarredona, T., Ieng, S. H., Benosman, R. B., and Linares-Barranco, B. (2014). On the use of orientation filters for 3d reconstruction in event-driven stereo vision. *Front. Neurosci.* 8:48. doi: 10.3389/fnins.2014.00048

Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R. J., and Delbruck, T. (2009). "A pencil balancing robot using a pair of aer dynamic vision sensors," in *IEEE International Symposium on Circuits and Systems (ISCAS)* (Taipei), 781–784.

Corneil, D. S., Neftci, E., Indiveri, G., and Pfeiffer, M. (2014). "Learning, inference, and replay of hidden state sequences in recurrent spiking neural networks," in *Computational and Systems Neuroscience (COSYNE)* (Salt Lake City, UT), 1–2.

Coultrip, R., Granger, R., and Lynch, G. (1992). A cortical model of winner-take-all competition via lateral inhibition. *Neural Netw.* 5, 47–54. doi: 10.1016/S0893-6080(05)80006-1

Delbruck, T., Linares-Barranco, B., Culurciello, E., and Posch, C. (2010). "Activity-driven, event-based vision sensors," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)* (Paris), 2426–2429. doi: 10.1109/ISCAS.2010.5537149

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. B* 39, 1–38.

Douglas, R. J., Mahowald, M. A., and Martin, K. A. C. (1994). "Hybrid analog-digital architectures for neuromorphic systems," in *Proceedings of 1994 IEEE World Congress on Computational Intelligence* (Orlando, FL), 1848–1853.

Doya, K., Samejima, K., Katagiri, K.-I., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Comput.* 14, 1347–1369. doi: 10.1162/089976602753712972

Farabet, C., Paz, R., Pérez-Carrasco, J., Zamarreño-Ramos, C., Linares-Barranco, A., LeCun, Y., et al. (2012). Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing. *Front. Neurosci.* 6:32. doi: 10.3389/fnins.2012.00032

Farhang-Boroujeny, B. (2013). *Adaptive Filters: Theory and Applications*. Hoboken, NJ: John Wiley & Sons. doi: 10.1002/9781118591352

Fukushima, K. (1980). Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* 36, 193–202.

Gabor, D. (1946). Theory of communication. *J. IEEE* 93, 429–459.

Galluppi, F., Lagorce, X., Stromatias, E., Pfeiffer, M., Plana, L. A., Furber, S. B., et al. (2014). A framework for plasticity implementation on the SpiNNaker neural architecture. *Front. Neurosci.* 8:429. doi: 10.3389/fnins.2014.00429

Gollisch, T., and Meister, M. (2008). Rapid neural coding in the retina with relative spike latencies. *Science* 319, 1108–1111. doi: 10.1126/science.1149639

Haruno, M., Wolpert, D., and Kawato, M. (2001). MOSAIC model for sensorimotor learning and control. *Neural Comput.* 13, 2201–2220. doi: 10.1162/089976601750541778

Huang, L., Shimizu, A., and Kobatake, H. (2004). "Classification-based face detection using Gabor filter features," in *Proceedings of Sixth IEEE International Conference on Automatic Face and Gesture Recognition* (Seoul), 397–402.

Hubel, D. H., and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* 160, 106–154. doi: 10.1113/jphysiol.1962.sp006837

Ilonen, J., Kamarainen, J.-K., and Kalviainen, H. (2007). "Fast extraction of multi-resolution gabor features," in *14th International Conference on Image Analysis and Processing (ICIAP)* (Modena), 481–486. doi: 10.1109/ICIAP.2007.4362824

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Comput.* 3, 79–87. doi: 10.1162/neco.1991.3.1.79

Jaeger, H. (2002). *Tutorial on Training Recurrent Neural Networks, Covering bppt, rtrl, ekf and the "Echo State Netwrok" Approach.* Technical report, German National Research Center for Information Technology.

Jaeger, H., and Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* 304, 78–80. doi: 10.1126/science.1091277

Jordan, M. I., and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Comput.* 6, 181–214. doi: 10.1162/neco.1994.6.2.181

Kappel, D., Nessler, B., and Maass, W. (2014). STDP installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLoS Comput. Biol.* 10:e1003511. doi: 10.1371/journal.pcbi.1003511

Lee, J. H., Delbruck, T., Pfeiffer, M., Park, P. K., Shin, C.-W., Ryu, H., et al. (2014). Real-time gesture interface based on event-driven processing from stereo silicon retinas. *IEEE Trans. Neural Netw. Learn. Syst.* 25, 2250–2263. doi: 10.1109/TNNLS.2014.2308551

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128X128 120dB 15us latency asynchronous temporal contrast vision sensor. *IEEE J. Solid State Circ.* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Lin, C., and Huang, C. (2005). "A complex texture classification algorithm based on gabor-type filtering cellular neural networks and self-organized fuzzy inference neural networks," in *IEEE International Symposium on Circuits and Systems (ISCAS)* (Kobe), 3942–3945.

Liu, S.-C., and Oster, M. (2006). "Feature competition in a spike-based winner-take-all VLSI network," in *Proceedings of 2006 IEEE International Symposium on Circuits and Systems (ISCAS)* (Kos), 3634–3637.

Lowe, D. G. (1999). "Object recognition from local scale-invariant features," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)* (Kerkyra), 1150–1157.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* 60, 91–110. doi: 10.1023/B:VISI.0000029664.99615.94

Maass, W., and Markram, H. (2004). On the computational power of circuits of spiking neurons. *J. Comput. syst. Sci.* 69, 593–616. doi: 10.1016/j.jcss.2004.04.001

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* 14, 2531–2560. doi: 10.1162/089976602760407955

Mead, C., and Mahowald, M. (1988). A silicon model of early visual processing. *Neural Netw.* 1, 91–97. doi: 10.1016/0893-6080(88)90024-X

Mueggler, E., Huber, B., and Scaramuzza, D. (2014). "Event-based, 6-DOF pose tracking for high-speed maneuvers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2761–2768.

Mutch, J., Knoblich, U., and Poggio, T. (2010). *CNS: a GPU-Based Framework for Simulating Cortically-Organized Networks*. Cambridge, MA: CBCL 286, MIT Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Nessler, B., Pfeiffer, M., Buesing, L., and Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput. Biol.* 9:e1003037. doi: 10.1371/journal.pcbi.1003037

Nessler, B., Pfeiffer, M., and Maass, W. (2009). "STDP enables spiking neurons to detect hidden causes of their inputs," in *Proceedings of Neural Information Processing Systems (NIPS)* (Vancouver, BC), 1357–1365.

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* 7:178. doi: 10.3389/fnins.2013.00178.

Olshausen, B. A., and Field, D. J. (1997). Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vis. Res.* 37, 3311–3326. doi: 10.1016/S0042-6989(97)00169-7

Oster, M., Douglas, R. J., and Liu, S.-C. (2009). Computation with spikes in a winner-take-all network. *Neural Comput.* 21, 2437–2465. doi: 10.1162/neco.2009.07-08-829

Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS. *IEEE J. Solid State Circ.* 46, 259–275. doi: 10.1109/JSSC.2010.2085952

Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., and Delbruck, T. (2014). Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *Proc. IEEE* 102, 1470–1484. doi: 10.1109/JPROC.2014.2346153

Rahimi Azghadi, M., Iannella, N., Al-Sarawi, S. F., Indiveri, G., and Abbott, D. (2014). Spike-based synaptic plasticity in silicon: design, implementation, application, and challenges. *Proc. IEEE* 102, 717–737. doi: 10.1109/JPROC.2014.2314454

Riesenhuber, M., and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nat. Neurosci.* 11, 1019–1025.

Roska, B., and Werblin, F. (2003). Rapid global shifts in natural scenes block spiking in specific ganglion cell types. *Nat. Neurosci.* 6, 600–608. doi: 10.1038/nn1061

Schmidhuber, J. (1991). Learning factorial codes by predictability minimization. *Neural Comput.* 4, 863–879. doi: 10.1162/neco.1992.4.6.863

Schrauwen, B., Verstraeten, D., and Campenhout, J. V. (2007). "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Sympsosium on Artificial Neural Networks* (Bruges), 471–482.

Serre, T., Bileschi, S., Wolf, L., Riesenhuber, M., and Poggio, T. (2006). Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 2007. doi: 10.1109/TPAMI.2007.56

Serre, T., Riesenhuber, M., Louie, J., and Poggio, T. (2002). "On the role of object-specific features for real world object recognition in biological vision," in *Proceedings of Biologically Motivated Computer Vision* (Tübingen), 387–397.

Sheik, S., Pfeiffer, M., Stefanini, F., and Indiveri, G. (2013). "Spatio-temporal spike pattern classification in neuromorphic systems," in *Biomimetic and Biohybrid Systems*, eds N. F. Lepora, A. Mura, H. G. Krapp, P. F. M. J. Verschure, and T. J. Prescott (London: Springer), 262–273.

Tapson, J. C., Cohen, G. K., Afshar, S., Stiefel, K. M., Buskila, Y., Wang, R. M., et al. (2013). Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Front. Neurosci.* 7:153. doi: 10.3389/fnins.2013.00153.

Uchibe, E., and Doya, K. (2004). "Competitive-cooperative-concurrent reinforcement learning with importance sampling," in *Proceedings of International Conference on Simulation of Adaptive Behavior: From Animals and Animats* (Los Angeles, CA), 287–296.

Uzzell, U. J., and Chichilnisky, E. J. (2004). Precision of spike trains in primate retinal ganglion cells. *J. Neurophysiol.* 92, 780–789. doi: 10.1152/jn.01171.2003

Wallis, G., and Bülthoff, H. (1999). Learning to recognize objects. *Trends Cogn. Sci.* 3, 22–31. doi: 10.1016/S1364-6613(98)01261-3

Wallis, G., and Rolls, E. (1997). A model of invariant object recognition in the visual system. *Prog. Neurobiol.* 51, 167–194. doi: 10.1016/S0301-0082(96)00054-8

Yuksel, S. E., Wilson, J. N., and Gader, P. D. (2012). Twenty years of mixture of experts. *IEEE Trans. Neural Netw. Learn. Syst.* 23, 1177–1193. doi: 10.1109/TNNLS.2012.2200299

Conflict of Interest Statement: The Reviewer Federico Corradi declares that, despite being affiliated to the same institution as author Michael Pfeiffer, the review process was handled objectively and no conflict of interest exists. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 20 October 2014; accepted: 03 February 2015; published online: 24 February 2015.

Citation: Lagorce X, Ieng S-H, Clady X, Pfeiffer M and Benosman RB (2015) Spatiotemporal features for asynchronous event-based data. Front. Neurosci. ***9****:46. doi: 10.3389/fnins.2015.00046*

This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.

ORIGINAL RESEARCH
published: 29 April 2015
doi: 10.3389/fnins.2015.00141

A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses

Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska and Giacomo Indiveri *

Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland

OPEN ACCESS

Edited by:
Gert Cauwenberghs, University of California, San Diego, USA

Reviewed by:
Siddharth Joshi, University of California, San Diego, USA
Srinjoy Das, University of California, San Diego, USA

Correspondence:
Giacomo Indiveri, Institute of Neuroinformatics, University of Zurich and ETH Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland
giacomo@ini.uzh.ch

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *21 October 2014*
Accepted: *06 April 2015*
Published: *29 April 2015*

Citation:
Qiao N, Mostafa H, Corradi F, Osswald M, Stefanini F, Sumislawska D and Indiveri G (2015) A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. Front. Neurosci. 9:141. doi: 10.3389/fnins.2015.00141

Implementing compact, low-power artificial neural processing systems with real-time on-line learning abilities is still an open challenge. In this paper we present a full-custom mixed-signal VLSI device with neuromorphic learning circuits that emulate the biophysics of real spiking neurons and dynamic synapses for exploring the properties of computational neuroscience models and for building brain-inspired computing systems. The proposed architecture allows the on-chip configuration of a wide range of network connectivities, including recurrent and deep networks, with short-term and long-term plasticity. The device comprises 128 K analog synapse and 256 neuron circuits with biologically plausible dynamics and bi-stable spike-based plasticity mechanisms that endow it with on-line learning abilities. In addition to the analog circuits, the device comprises also asynchronous digital logic circuits for setting different synapse and neuron properties as well as different network configurations. This prototype device, fabricated using a 180 nm 1P6M CMOS process, occupies an area of 51.4 mm^2, and consumes approximately 4 mW for typical experiments, for example involving attractor networks. Here we describe the details of the overall architecture and of the individual circuits and present experimental results that showcase its potential. By supporting a wide range of cortical-like computational modules comprising plasticity mechanisms, this device will enable the realization of intelligent autonomous systems with on-line learning capabilities.

Keywords: spike-based learning, Spike-Timing Dependent Plasticity (STDP), real-time, analog VLSI, Winner-Take-All (WTA), attractor network, asynchronous, brain-inspired computing

1. Introduction

Recent advances in neural network modeling and theory, combined with advances in technology and computing power, are producing impressive results in a wide range of application domains. For example, large-scale deep-belief neural networks and convolutional networks now represent the state-of-the-art for speech recognition and image segmentation applications (Mohamed et al., 2012; Farabet et al., 2013). However, the mostly sequential and synchronous clocked nature of conventional computing platforms is not optimally suited for the implementation of these types of massively parallel neural network architectures. For this reason a new generation of custom neuro-computing hardware systems started to emerge. These systems are typically composed of custom Very Large Scale Integration (VLSI) chips that either contain digital processing cores with dedicated memory structures and communication schemes optimized for spiking neural networks architectures (Wang et al., 2013; Furber et al., 2014; Neil and Liu, 2014), or full-custom digital circuit solutions that implement large arrays of spiking neurons with programmable synaptic connections (Merolla et al., 2014). While these devices and systems have high potential for solving machine learning tasks and applied research problems, they do not emulate directly the dynamics of real neural systems.

At the other end of the spectrum, neuromorphic engineering researchers have been developing hardware implementations of detailed neural models, using mixed signal analog-digital circuits to reproduce faithfully neural and synaptic dynamics, in a basic research effort to understand the principles of neural computation in physical hardware systems (Douglas et al., 1995; Liu et al., 2002; Chicca et al., 2014). By studying the physics of computation of neural systems, and reproducing it through the physics of transistors biased in the subthreshold regime (Liu et al., 2002), neuromorphic engineering seeks to emulate biological neural computing systems efficiently, using the least amount of power and silicon real-estate possible. Examples of biophysically realistic neural electronic circuits built following this approach range from models of single neurons (Mahowald and Douglas, 1991; Farquhar and Hasler, 2005; Hynna and Boahen, 2007; van Schaik et al., 2010), to models of synaptic dynamics (Liu, 2003; Bartolozzi and Indiveri, 2007a; Xu et al., 2007), to auditory/visual sensory systems (Sarpeshkar et al., 1996; van Schaik and Meddis, 1999; Zaghloul and Boahen, 2004; Costas-Santos et al., 2007; Liu and Delbruck, 2010), to reconfigurable spiking neural network architectures with learning and plasticity (Giulioni et al., 2008; Hsieh and Tang, 2012; Ramakrishnan et al., 2012; Yu et al., 2012; Chicca et al., 2014).

In this paper we propose to combine the basic research efforts with the applied research ones, by presenting a VLSI architecture that can be used to both carry out research experiments in computational neuroscience, and to develop application solutions for practical tasks. The architecture proposed comprises electronic neuromorphic circuits that directly emulate the physics of real neurons and synapses to faithfully reproduce their adaptive and dynamic behavior, together with digital logic circuits that can set both the properties of the individual synapse and neuron elements as well as the topology of the neural network. In particular, this architecture has been developed to implement spike-based adaptation and plasticity mechanisms, and to carry out on-chip on-line learning for tasks that require the system to adapt to the changes in the environment it interacts with. Given these characteristics, including the ability to arbitrarily reconfigure the network topology also at run-time, we named this device the Reconfigurable On-line Learning Spiking Neuromorphic Processor (ROLLS neuromorphic processor).

The main novelty of the work proposed, compared to previous analogous approaches (Indiveri et al., 2006; Giulioni et al., 2008; Ramakrishnan et al., 2012; Yu et al., 2012) consists in the integration of analog bi-stable learning synapse circuits with asynchronous digital logic cells and in the embedding of these mixed-signal blocks in a large multi-neuron architecture. The combination of analog and digital circuits, with both analog and digital memory elements, within the same block provides the device with an important set of programmable features, including the ability to configure arbitrary network connectivity schemes. At the analog circuit design level, we present improvements in the neuron and spike-based learning synapses over previously proposed ones (Indiveri et al., 2011; Chicca et al., 2014), which extend their range of behaviors and significantly reduce device mismatch effects. At the system application level we demonstrate, for the first time, both computational neuroscience models of attractor networks and image classification neural networks implemented exclusively on custom mixed-signal analog-digital neuromorphic hardware, with no extra pre- or post-processing done in software. In the next section we describe the ROLLS neuromorphic processor system-level block diagram, highlighting its dynamic and spike-based learning features. In Section 2.2 we describe in detail the circuits that are present in each building block, and in Section 3 we present system level experimental results showcasing examples of both computational neuroscience models and machine vision pattern recognition tasks. Finally, in Sections 4, 5 we discuss the results obtained and summarize our contribution with concluding remarks.

2. Materials and Methods

2.1. The Neuromorphic Processor Architecture

The block-diagram of the ROLLS neuromorphic processor architecture is shown in **Figure 1**. The device comprises a configurable array of synapse circuits that produce biologically realistic response properties and spiking neurons that can exhibit a wide range of realistic behaviors. Specifically, this device comprises a row of 256×1 silicon neuron circuits, an array of 256×256 learning synapse circuits for modeling long-term plasticity mechanisms, an array of 256×256 programmable synapses with short-term plasticity circuits, a 256×2 row of linear integrator filters denoted as "virtual synapses" for modeling excitatory and inhibitory synapses that have shared synaptic weights and time constants, and additional peripheral analog/digital Input/Output (I/O) circuits for both receiving and transmitting spikes in real-time off-chip.

The ROLLS neuromorphic processor was fabricated using a standard 180 nm Complementary Metal-Oxide-Semiconductor (CMOS) 1P6M process. It occupies an areas of 51.4 mm^2 and has approximately 12.2 million transistors. The die photo of the chip is shown in **Figure 2**. The area distribution of main circuit blocks is shown in **Table 1**. The silicon neurons contain circuits that implement a model of the adaptive exponential Integrate-and-Fire (I&F) neuron (Brette and Gerstner, 2005), post-synaptic learning circuits used to implement the spike-based weight-update/plasticity mechanism in the array of long-term plasticity synapses, and analog circuits that model homeostatic synaptic scaling mechanisms operating on very long time scales (Rovere et al., 2014). The array of long-term plasticity synapses comprises pre-synaptic spike-based learning circuits

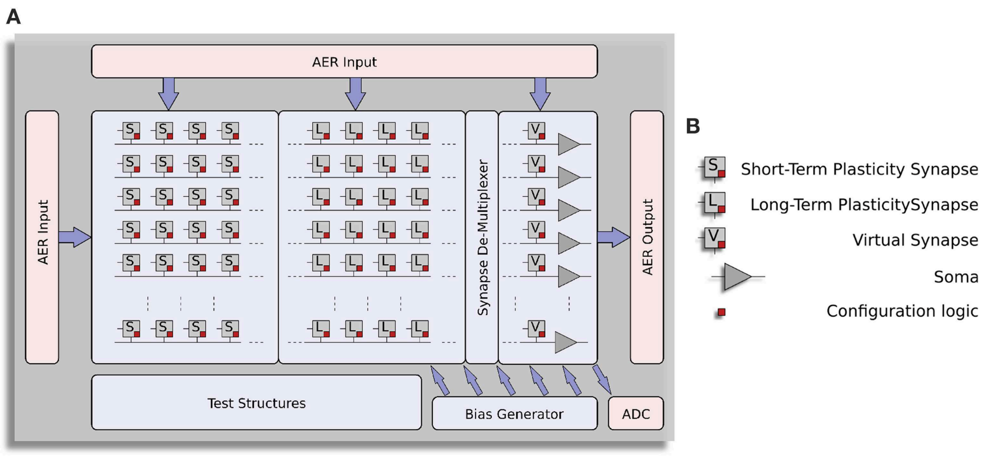


FIGURE 1 | Architecture of ROLLS neuromorphic processor. (A) Block diagram of the architecture, showing two distinct synapse arrays (short-term plasticity and long-term plasticity synapses), an additional row of synapses (virtual synapses) and a row of neurons (somas). A synapse de-multiplexer block is used to connect the rows from the synapse arrays to the neurons (see main text for details). Peripheral circuits include asynchronous digital AER logic blocks, an Analog-to-Digital converter, and a programmable on-chip bias-generator. **(B)** Block-diagram legend.

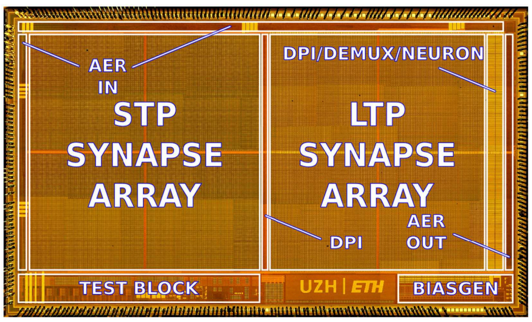


FIGURE 2 | Micro-photograph of the ROLLS neuromorphic processor. The chip was fabricated using a 180 nm CMOS process and occupies an area of 51.4 mm^2, comprising 12.2 million transistors.

TABLE 1 | Circuits area distribution.

Circuit	Dimensions ($\mu m \times \mu m$)	Number	Total area: (mm^2)	(%)
Neuron	55.69×16.48	256	0.235	0.47
Post-synaptic learning	39.09×16.48	256	0.165	0.32
LTP synapse	15.3×16.48	64 k	16.147	31.41
STP synapse	16.24×16.48	64 k	17.129	33.32
Virtual synapse	35.6×16.48	512	0.300	0.58
Synapse de-mux	49.56×4389.4	1	0.218	0.42
AER in (columns)	8770×154	1	0.135	0.26
AER in (rows)	112×4357	1	0.488	0.95
AER out	166.2×4274.9	1	0.710	1.38
BiasGen	539.5×1973	1	1.064	2.07

The remaining area used in the chip is occupied by the pads and additional test structures.

with bi-stable synaptic weights, that can undergo either Long-Term Potentiation (LTP) or Long-Term Depression (LTD), (see Section 2.1.2 for details). The array of Short-Term Plasticity (STP) synapses comprises synapses with programmable weights and STP circuits that reproduce short-term adaptation dynamics. Both arrays contain analog integrator circuits that implement faithful models of synaptic temporal dynamics (see Section 2.1.1). Digital configuration logic in each of the synapse and neuron circuits allows the user to program the properties of the synapses, the topology of the network, and the properties of the neurons.

The architecture comprises also a "synapse de-multiplexer" static logic circuit, which allows the user to choose how many rows of plastic synapses should be connected to the neurons. It is a programmable switch-matrix that configures the connectivity between the synapse rows and the neuron columns. By default, each of the 256 rows of 1×512 synapses is connected to its corresponding neuron. By changing the circuit control bits it is possible to allocate multiple synapse rows to the neurons, thereby disconnecting and sacrificing the unused neurons. In the extreme case all 256×512 synapses are assigned to a single neuron, and the remaining 255 neurons remain unused.

An on-chip programmable bias generator, optimized for subthreshold circuits (Delbruck et al., 2010) is used to set all of the bias currents that control the parameters of the synapses and neurons (such as time constants, leak currents, *etc.*).

An Analog to Digital Converter (ADC) circuit converts the subthreshold currents produced by selected synapse and neuron circuits into a stream of voltage pulses, using a linear pulse-frequency-modulation scheme, and transmits them off-chip as digital signals.

Finally, peripheral asynchronous I/O logic circuits are used for receiving input spikes and transmitting output ones, using the Address-Event Representation (AER) communication protocol (Deiss et al., 1998; Boahen, 2000).

2.1.1. Synapse Temporal Dynamics

In the ROLLS neuromorphic processor all synapses process input spikes in real-time, as they arrive. Similarly the neurons transmit the spikes they produce immediately, as they are generated. In these types of architectures time represents itself and input data is processed instantaneously. There is no virtualization of time and no mechanism for storing partial results in memory banks. As a consequence, the circuits must operate with time-constants that are well-matched to those of the signals they are designed to process. Since this device is intended to be used in behaving systems that interact with the environment in natural real-world scenarios, it is important to design circuits that can implement a wide range of time constants, including very slow, biologically plausible, ones. To achieve this, and to model neural dynamics with biologically plausible time constants, we used the Differential Pair Integrator (DPI) (Bartolozzi and Indiveri, 2007b). This is a current-mode log-domain integrator. When biased in the subthreshold regime, this circuit can obtain long time constants, even with relatively small and compact capacitors. For example, in the 180 nm technology used, with a capacitor of 1 pF, we could obtain time constants of the order of tens of milliseconds without resorting to any advanced design techniques. However, to realize even longer time constants (e.g., of the order of hundreds of milliseconds), we used a shifted-source biasing technique, as described in Linares-Barranco and Serrano-Gotarredona (2003).

The synapse circuits in the two synapse arrays of the ROLLS neuromorphic processor convert input voltage spikes into output currents which have non-linear dynamics, due to their adaptation or learning features. In addition, to model the synapse temporal dynamics, the currents produced by the circuit elements in the array are further integrated by a linear temporal filter. If we assume that all the synapses in an array have the same temporal dynamics (i.e., share the same time constants), then we can exploit Kirchhoff's current law and sum the output currents of all synapses in a row into a single DPI circuit. This allows us to save a significant amount of silicon real-estate, as we can use only one DPI per row, in each array. In particular, we use one excitatory DPI in the long-term plasticity array configured to produce time constants of the order of hundreds of milliseconds, to model the dynamics of N-Methyl-D-Aspartate (NMDA) receptors, and two DPI circuits (one for excitatory and one for inhibitory synaptic dynamics) in the STP array, configured with time constants of the order of tens of milliseconds, to model the dynamics of AMPA and GABA receptors, respectively.

We use the same principle for the 256×2 "virtual synapse" integrators in the architecture. These circuits comprise two DPI integrators per row (one for the excitatory synapse and one for the inhibitory one) with fixed sets of weights and shared time-constant parameters, biased to operate in their linear operating range. By time-multiplexing input spikes to a single virtual synapse we can model the effect of multiple independent inputs to the targeted neuron. For example, by stimulating the DPI with a single 10 KHz spike train, we can model the effect of 1000 synapses receiving a 10 Hz input spike train.

2.1.2. The Spike-Based Learning Algorithm

Many models of Spike-Timing Dependent Plasticity (STDP) have been proposed in the computational neuroscience literature (Abbott and Nelson, 2000; Markram et al., 2012). However, a growing body of evidence is revealing that learning algorithms based on spike-timing alone cannot account for all of the phenomenology observed neurophysiological experiments (Lisman and Spruston, 2010), have poor memory retention performance (Billings and van Rossum, 2009), and require additional mechanisms to learn both spike-time correlations and mean firing rates in the input patterns (Senn, 2002).

For this reason, we chose to implement the spike-driven synaptic plasticity rule proposed by Brader et al. (2007), which has been shown to reproduce many of the behaviors observed in biology, and has performance characteristics that make it competitive with the state-of-the-art machine learning methods (Brader et al., 2007). This algorithm does not rely on spike-timing alone. It updates the synaptic weights according to the timing of the pre-synaptic spike, the state of the post-synaptic neuron's membrane potential, and its recent spiking activity. It assumes that the synaptic weights are bounded, and that, on long time-scales, they converge to either a high state, or a low one. However, in order to avoid updating all synapses in exactly the same way, this algorithm requires a stochastic weight update mechanism (see Brader et al., 2007 for details).

The requirements and features of this algorithm make it particularly well-suited for neuromorphic hardware implementation: the bi-stability feature removes the problematic need of storing precise analog variables on long-time scales, while the probabilistic weight update requirement can be obtained by simply exploiting the variability in the input spike trains (typically produced by a Poisson process) and the variability in the post-synaptic neuron's membrane potential (typically driven by noisy sensory inputs).

The weight-update rule for a given synapse i is governed by the following equations, which are evaluated upon the arrival of each pre-synaptic spike:

$$\begin{cases} w_i = w_i + \Delta w^+ & \text{if } V_{mem}(t_{pre}) > \theta_{mem} \text{ and} \\ & \theta_1 < Ca(t_{pre}) < \theta_3 \\ w_i = w_i - \Delta w^- & \text{if } V_{mem}(t_{pre}) < \theta_{mem} \text{ and} \\ & \theta_1 < Ca(t_{pre}) < \theta_2 \end{cases} \tag{1}$$

where w_i represents an internal variable that encodes the bi-stale synaptic weight; the terms Δw^+ and Δw^- determine the amplitude of the variable instantaneous increases and decreases; $V_{mem}(t_{pre})$ represents the post-synaptic neuron's membrane potential at the time of arrival of the pre-synaptic spike, and θ_{mem} is a threshold term that determines whether the weight should be increased or decreased; the term $Ca(t_{pre})$ represents the post-synaptic neuron's Calcium concentration, which is proportional to the neuron's recent spiking activity, at the time of the pre-synaptic spike, while the terms θ_1, θ_2, and θ_3 are three thresholds that determine in which conditions the weights are allowed to be increased, decreased, or should not be updated. These "stop-learning" conditions are useful for normalizing the weights of all synapses afferent to the same neuron. They have been shown to be effective in extending the memory lifetime of recurrent spiking neural networks, and in increasing their capacity (Senn and Fusi, 2005).

In parallel to the instantaneous weight updates, the internal variable of the synapse w_i is constantly being driven toward one of two stable states, depending whether it is above or below a given threshold θ_w:

$$\begin{cases} \frac{d}{dt}w_i = +C_{drift} & \text{if } w_i > \theta_w \text{ and } w_i < w_{max} \\ \frac{d}{dt}w_i = -C_{drift} & \text{if } w_i < \theta_w \text{ and } w_i > w_{min} \end{cases} \quad (2)$$

where C_{drift} represents the rate at which the synapse is driven to its bounds, and w_{max} and w_{min} represent the high and low bounds, respectively. The actual weight J_i of the synapse i is a thresholded version of the internal variable w_i that is used to produce the Excitatory Post-Synaptic Current (EPSC) upon the arrival of the pre-synaptic spike:

$$J_i = J_{max} f(w_i, \theta_J) \quad (3)$$

where $f(x, \theta_J)$ can be a sigmoidal or hard-threshold function with threshold θ_J, and J_{max} is the maximum synaptic efficacy.

We will show in Section 2.2.3 experimental results that demonstrate how the circuits integrated in the ROLLS neuromorphic processor chip faithfully implement this learning algorithm.

2.2. The Neuromorphic Processor Building Blocks

Here we present the main building blocks used in the ROLLS neuromorphic processor chip, describing the circuit schematics and explaining their behavior.

2.2.1. The Silicon Neuron Block

The neuron circuit integrated in this chip is derived from the adaptive exponential I&F circuit proposed in Indiveri et al. (2011), which can exhibit a wide range of neural behaviors, such as spike-frequency adaptation properties, refractory period mechanism and adjustable spiking threshold mechanism. The circuit schematic is shown in **Figure 3**. It comprises an NMDA block ($M_{N1,N2}$), which implements the NMDA voltage gating function, a LEAK DPI circuit (M_{L1-L7}) which models the neuron's leak conductance, an AHP DPI circuit (M_{A1-A7}) in negative feedback mode, which implements a spike-frequency adaptation behavior, an Na$^+$ positive feedback block ($M_{Na1-Na5}$) which models the effect of Sodium activation and inactivation channels for producing the spike, and a K$^+$ block (M_{K1-K7}) which models the effect of the Potassium conductance, resetting the neuron and implementing a refractory period mechanism. The negative feedback mechanism of the AHP block, and the tunable reset potential of the K$^+$ block introduce two extra variables in the dynamic equation of the neuron that can endow it with a wide variety of dynamical behaviors (Izhikevich, 2003). As the neuron circuit equations are essentially the same of the adaptive I&F neuron model, we refer to the work of Brette and

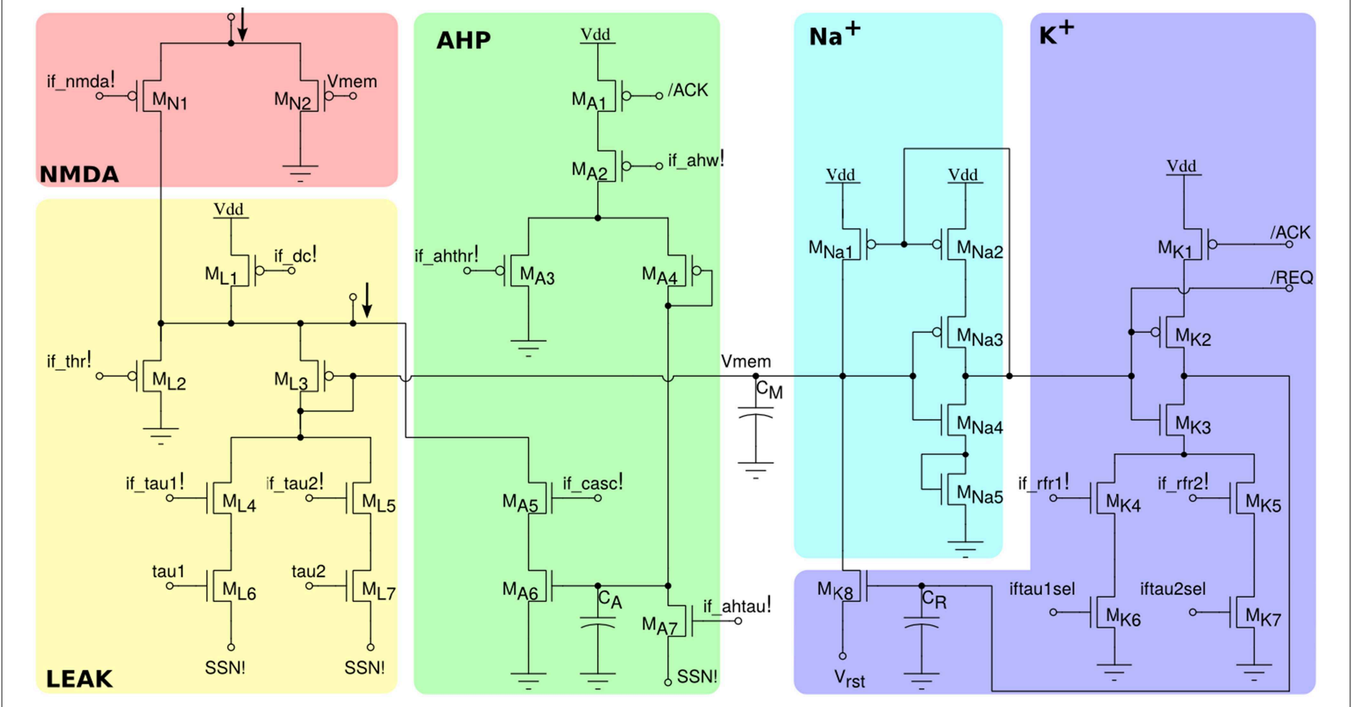


FIGURE 3 | Silicon neuron schematics. The NMDA block implements a voltage gating mechanism; the LEAK block models the neuron's leak conductance; the spike-frequency adaptation block AHP models the after-hyper-polarizing current effect; the positive-feedback block Na$^+$ models the effect of the Sodium activation and inactivation channels; reset block K$^+$ models the Potassium conductance functionality.

Gerstner (2005) for an extensive analysis of the repertoire of behaviors that this neuron model can reproduce, in comparison to, e.g., the Izhikevich neuron model.

All voltage bias variables in **Figure 3** ending with an exclamation mark represent global tunable parameters which can be precisely set by the on chip Bias Generator (BG). There are a total of 13 tunable parameters, which provide the user with high flexibility for configuring all neurons to produce different sets of behaviors. In addition, by setting the appropriate bits of the relative latches in each neuron, it is possible to configure two different leak time constants (if_tau1! / if_tau2!) and refractory period settings (if_rfr1! / if_rfr2!). This gives the user the opportunity to model up to four different types/populations of neurons within the same chip, that have different leak conductances and/or refractory periods.

An example of the possible behaviors that can be expressed by the silicon neuron are shown in **Figure 4**. The top-left quadrant shows measured data from the chip representing the neuron membrane potential in response to a constant current injection for different values of reset voltage. The top-right quadrant shows the neuron response to a constant current injection for different settings of its refractory period. The bottom-left quadrant demonstrates the spike-frequency adaptation behavior, obtained by appropriately tuning the relevant parameters in the AHP block of **Figure 3** and stimulating the neuron with a constant injection current. By further increasing the gain of the AHP negative feedback block the neuron can produce bursting behavior (see bottom-right quadrant of **Figure 4**).

Figure 5 shows the F-I curve of all neurons in the ROLLS neuromorphic processor (i.e., their firing rate as a function of the input injection current). The plot shows their average firing rate in solid line, and their standard deviation in the shaded area. The overall mismatch in the circuit, responsible for these deviations, is extremely small, if compared to other analog VLSI implementations of neural systems (Indiveri et al., 2006; Petrovici et al., 2014; Schmuker et al., 2014). The average value obtained from the measurement results of **Figure 5** is only 9.4%. The reason for this improvement lies in the increased size of some critical transistors in the soma circuit—major contributor to neuron's mismatch. For example, the M_{L4} and M_{L5} Field-Effect Transistors (FETs) that set the neuron's leak time constants are of (W/L) size of ($2\,\mu$m/$4\,\mu$m) , while M_{Na3} and M_{Na4}, responsible for the firing threshold are of size ($4\,\mu$m/$0.4\,\mu$m) and ($1\,\mu$m/$4\,\mu$m), respectively.

In addition to the neuron soma circuit, this block contains also post-synaptic plasticity circuits that are necessary for evaluating the weight update and "stop-learning" conditions described in Section 2.1.2. In particular these circuits integrate

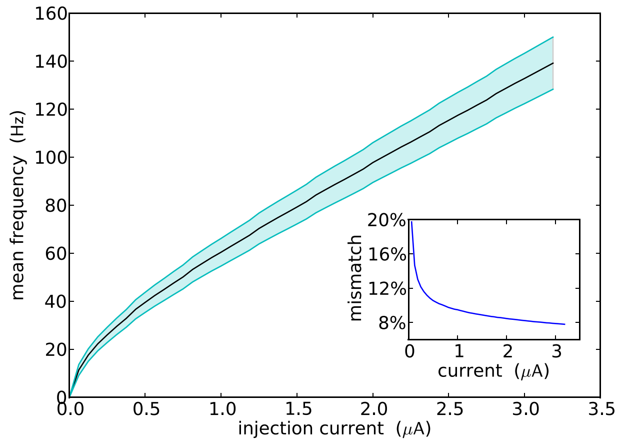


FIGURE 5 | Population response of all neurons in the array to constant injection currents. The variance in the measurements is due to device mismatch effects in the analog circuits.

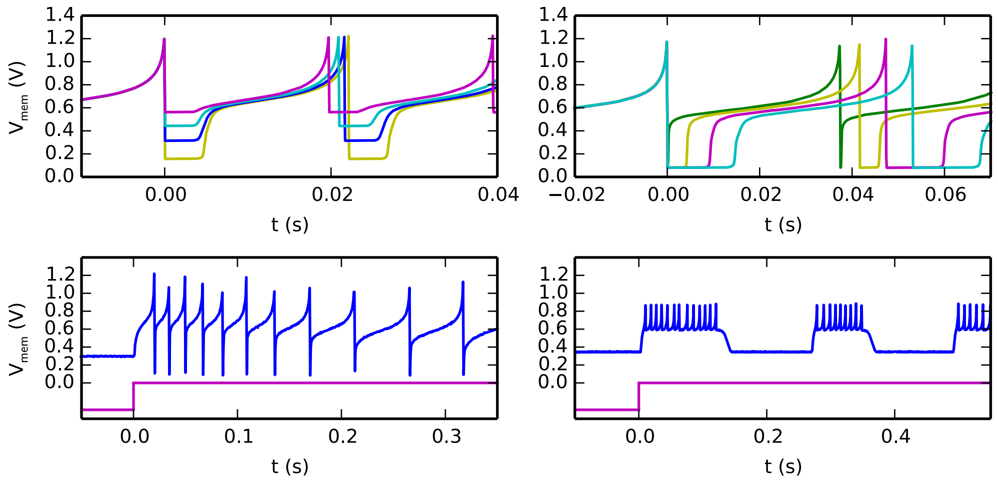


FIGURE 4 | Different biologically plausible neuron's behaviors: (top-left) membrane potential with tunable reset potential (different colors represent different reset potential settings), (top-right) membrane potential with tunable refractory period duration (different colors represent different refractory period settings), (bottom-left) neuron's spike-frequency adaptation behavior: the top trace represents the membrane potential and the bottom one represents the input current, (bottom-right) neuron's bursting behavior.

the spikes produced by the neuron into a current that models the neuron's Calcium concentration, and compare this current to three threshold currents that correspond to θ_1, θ_2, and θ_3 of Equation (1). In parallel, the neuron's membrane current (which is equivalent to the membrane potential in the theoretical model) is compared to an additional threshold equivalent to θ_{mem} of Equation (1). The schematic diagram of this circuit is shown in **Figure 6**. The post-synaptic neuron's Calcium concentration is computed using the DPI M_{D1-D5}; the comparisons with the fixed thresholds are made using three current-mode Winner-Take-All (WTA) circuits M_{W1-W9}, $M_{WU1-WU12}$, and $M_{WD1-WD12}$. The digital outcomes of these comparisons set the signals slnup and sldn which are then buffered and transmitted in parallel to all synapses afferent to this neuron belonging to the long-term plasticity array.

2.2.2. The Long-Term Plasticity Synapse Array

Each of the 256×256 synapse circuits in the long-term plasticity array comprises event-based programmable logic circuits for configuring both synapse and network properties, as well as analog/digital circuits for implementing the learning algorithm of Section 2.1.2. **Figure 7** shows both digital and analog circuit blocks. The digital logic part, shown in **Figure 7A** has an pulse generator circuit that manages the handshaking signals required by the AER protocol, and three one-bit configurable latches: one latch sets/resets the MON_EN signal, which enables/disables the synapse monitor circuit, which buffers the synapse weight V_w signal for off-chip reading. The remaining two latches are used to set the BC_EN and REC_EN signals, which control the activation modes of the synapse. There are three different activation modes can be configured: direct activation, broadcast activation and recurrent activation. **Figure 7B** shows a timing diagram in which the relative latches for enabling broadcast and recurrent activation modes are configured in a synapse, using a 4-phase handshaking protocol. In the direct activation mode the synapse is stimulated by an AER event that has the matching row and column address. In the broadcast activation mode the synapse is stimulated by an AER broadcast event (that has a dedicated address word) which targets the matching column address. All synapses belonging to the same column that have the BC_EN bit set high get stimulated in parallel, when the matching broadcast event is received. In the recurrent activation mode the synapse of column *j* is stimulated when the on-chip post-synaptic neuron of row *j* spikes. Therefore, it is possible to connect, internally, neuron *i* to neuron *j* by setting the REC_EN bit high of the synapse in row *i* and column *j*. In addition to these circuits, there is a pulse extender circuit which can increase the duration of the input pulse from nano-seconds to hundreds of micro-seconds.

The schematic diagram of the analog/digital weight update circuits is shown in **Figure 7C**. These circuits are subdivided into four sub-blocks: the SET block can be used to set/reset

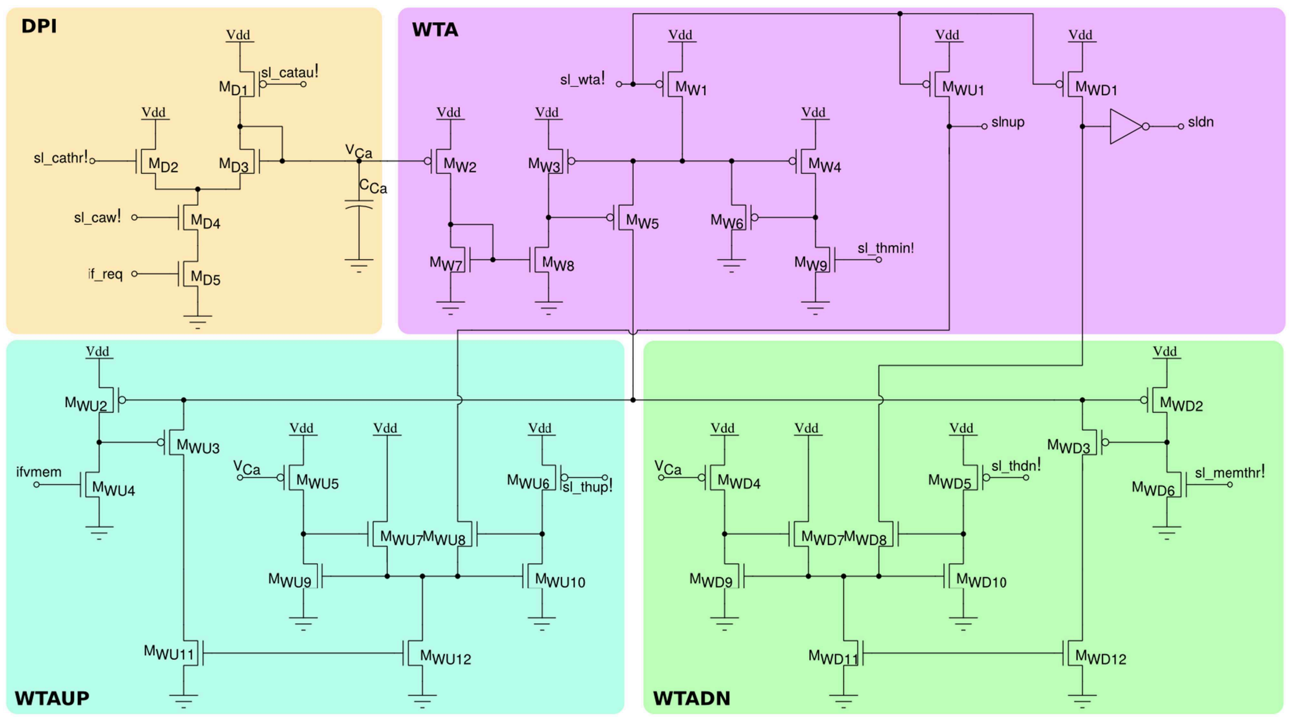


FIGURE 6 | Post-synaptic learning circuits for evaluating the algorithm's weight update and "stop-learning" conditions. The DPI circuit M_{D1-5} integrates the post-synaptic neuron spikes and produces a current proportional to the neuron's Calcium concentration. Three current-mode winner-take-all circuits *WTA*, *WTAUP*, and *WTADN* compare the Calcium concentration current to three set thresholds sl_thmin! , sl_thdn!, and sl_thup!, while the neuron's membrane current is compared to the threshold sl_memthr!.

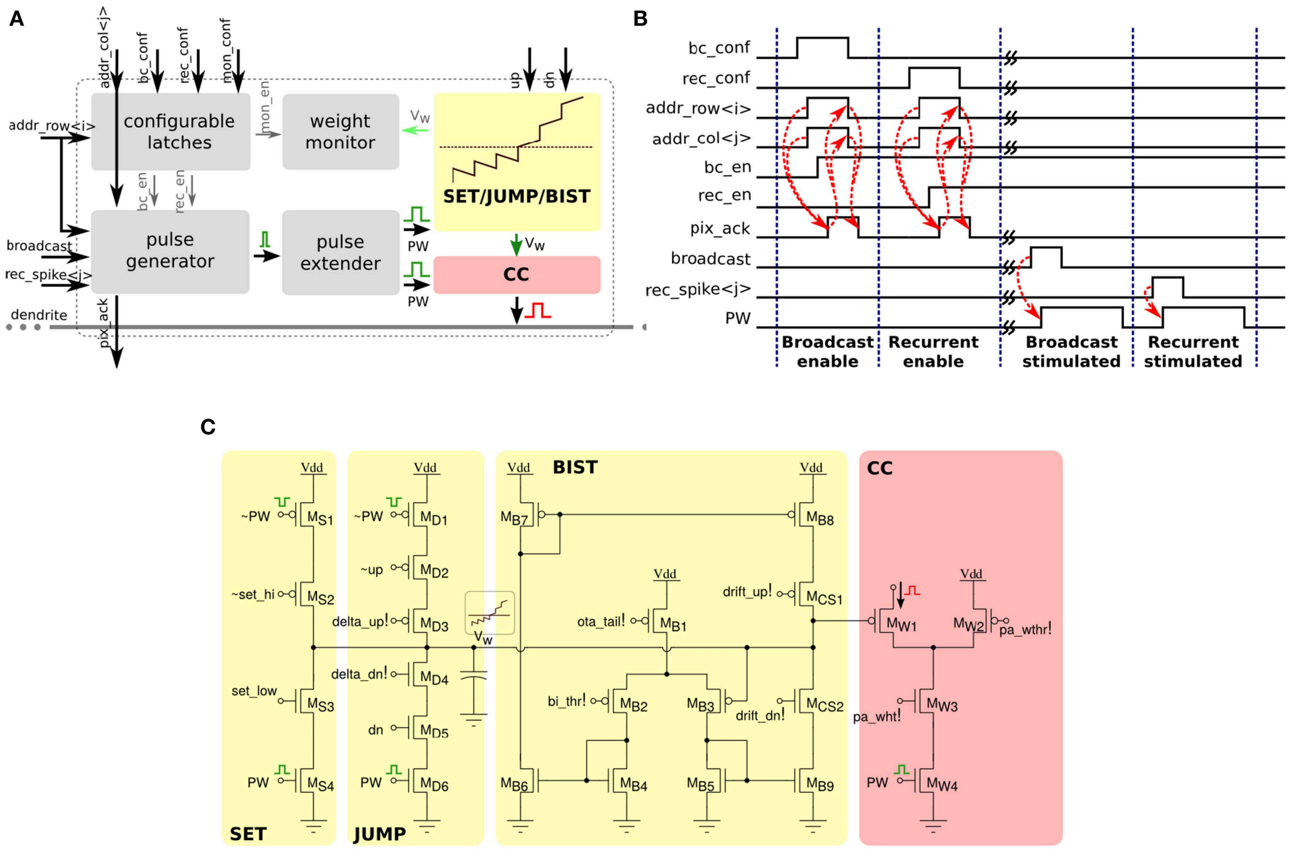


FIGURE 7 | Long-term plasticity synapse array element. (A) Plastic synapse configuration logic block diagram. **(B)** Timing diagram for broadcast and recurrent activation modes in one synapse using 4-phase handshaking protocol. Dashed red lines show the sequence between signals. **(C)** Schematic diagram of the bi-stable weight update and current generator blocks.

the bistable state of the synaptic weight by sending an AER event with the matching address and properly asserting the configuration signals set_hi and set_low. The JUMP block increases or decreases the synaptic weight internal variable (i.e., the voltage V_w) depending on the digital signals up and dn, that are buffered copies of the ones generated in the silicon neuron stop-learning block (see Section 2.2.1). The heights of the up and down jumps can be set by changing the delta_up! and delta_dn! signals. The BIST block consists of a wide-range transconductance amplifier configured in positive feedback mode, to constantly compare the V_w node with the threshold bi_thr!: if $V_w >$ bi_thr! then the amplifier slowly drives the V_w node, drifting toward the positive rail, otherwise it actively drives it toward the ground. The drift rates to the two states can be tuned by biases drift_up! and drift_dn!, respectively. The current converter (CC) block converts the V_w voltage into a thresholded EPSC with maximum amplitude set by pa_wht!.

Figure 8 shows experimental results that highlight the features of both synapse and neuron learning circuits in action: weight updates are triggered when the pre-synaptic spikes arrive, and when the post-synaptic neuron's Calcium concentration is in the appropriate range. Depending on the value of the Calcium concentration signal, the digital up and dn signal turn on or off. The weight internal variable is increased or decreased depending on where the membrane potential is with respect to the membrane threshold (see highlighted weight updates at $t = 273$ and $t = 405$). This variable is actively driven to the low or high bounds, depending if it is below or above the weight hreshold.

2.2.3. The Short-Term Plasticity Synaptic Array

The array of STP synapses contains circuits that allow users to program the synaptic weights, rather than changing them with a fixed on-chip learning algorithm. Specifically, each synapse has a two-bit programmable latch that can be used to set one of four possible weight values. In addition, it has an extra latch that can set the type of synapse (excitatory or inhibitory). In the excitatory mode, the synapse has additional circuits for modeling Short-Term Depression (STD) dynamics (Rasche and Hahnloser, 2001; Boegerhausen et al., 2003) whereby the magnitude of the EPSC decreases with every input spike, and recovers slowly in absence of inputs. **Figure 9** shows both a block diagram of all synapse components, and the schematic diagram of the synapse analog circuits. In addition to the latches for setting the weight,

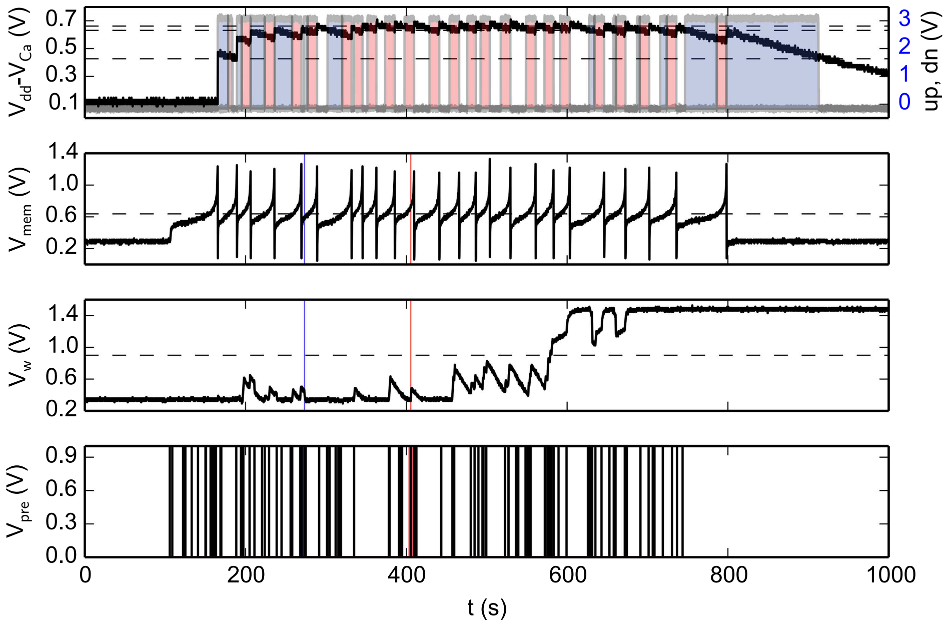


FIGURE 8 | Spike-based learning circuit measurements. The bottom trace represents the pre-synaptic input spikes; the second trace from the bottom represents the bi-stable internal variable (node V_W of **Figure 7**); the third trace represents the post-synaptic neuron's membrane potential; and the top trace shows both a voltage trace proportional to the neuron's integrated spiking activity as well as the digital control signals that determine whether to increase (red shaded area), decrease (blue shaded area) or leave V_W unchanged (no shaded area). The horizontal lines represent the thresholds used in the learning algorithm (see Section 2.1.2), while the vertical lines at $t = 273$ s (blue line) and $t = 405$ s (red line) are visual guides to show where the membrane potential is, with respect to its threshold, for down and up jumps in V_W respectively.

there are two extra latches for configuring the synapse activation mode. As for the long-term-plasticity synapses, there are three possible activation modes: direct, broadcast, and recurrent (see Section 2.2.2).

The left panel of **Figure 9B** shows the excitatory CC and the STD circuit. The CC at the top generates a current that is proportional to the 2-bit weight. The proportionality constant is controlled through analog biases. This current charges up the C_{STD} capacitor through the diode connected p-FET M_{S3} so that at steady state, the gate voltages of M_{S1} and M_{W2} are equal. A pre-synaptic pulse on the *PW* port activates the I_{exc} current branch, and produces a current that initially is proportional to the 2-bit weight original current. At the same time, the *PW* pulse activates also the STD branch through transistor M_{S5} and an amount of positive charge that is controlled by the bias *STD* is removed from the capacitor C_{STD}. The gate voltage of M_{W2} is now momentarily lower than that of M_{S1}, and recovers slowly through the diode connected p-FET M_{S3}. Pulses that arrive before the capacitor voltage has recovered completely will generate a current that is smaller than the original one, and will further depress the effective synaptic weight through the STD branch. The excitatory block is only active if the E/I voltage is high. If E/I is low, the inhibitory current DAC in the right panel of **Figure 9B** is active and generates a weight-proportional inhibitory current on *PW* pulses.

Figure 10 illustrates how the STD behavior in the synapse: a spike burst was used to activate a programmable synapse. This resulted in a drop in synaptic efficacy during the later part of the burst. During a period of no stimulation the synapse recovered and responded with large Excitatory Post-Synaptic Potentials (EPSPs) to the initial part of the following burst, before depressing again. The responses to the two bursts are not identical in **Figure 10** as the state of the neuron, synapse, and DPI circuits are not exactly the same at the onset of each burst.

2.2.4. The Peripheral Input/Output Blocks

The peripheral digital circuits are used to transmit signals into and out of the chip. Given the real-time nature of our system, we use asynchronous digital circuits and quasi-delay-insensitive circuit design techniques (Manohar, 2006) to avoid discretization or virtualization of time. The AER communication protocol used encodes signals as the address of the destination synapse or as a control word for the input side, and as the address of the sender neuron in the output circuits.

2.2.4.1. AER input circuits

Input spike events as well as chip configuration events are sent through a common input interface that uses a 21-bit address space. Input addresses are decoded into a total of 1,249,553 possible patterns subdivided into three categories: *Addressing*, *Local configuration*, and *Global configuration*. *Addressing* inputs are decoded into a row and column address and are interpreted as a spike Address-Event (AE), which are sent to the desired target synapse of a target neuron. *Local configuration* AEs contain the row and column address of the target element as well as extra configuration bits that are written to the local latches

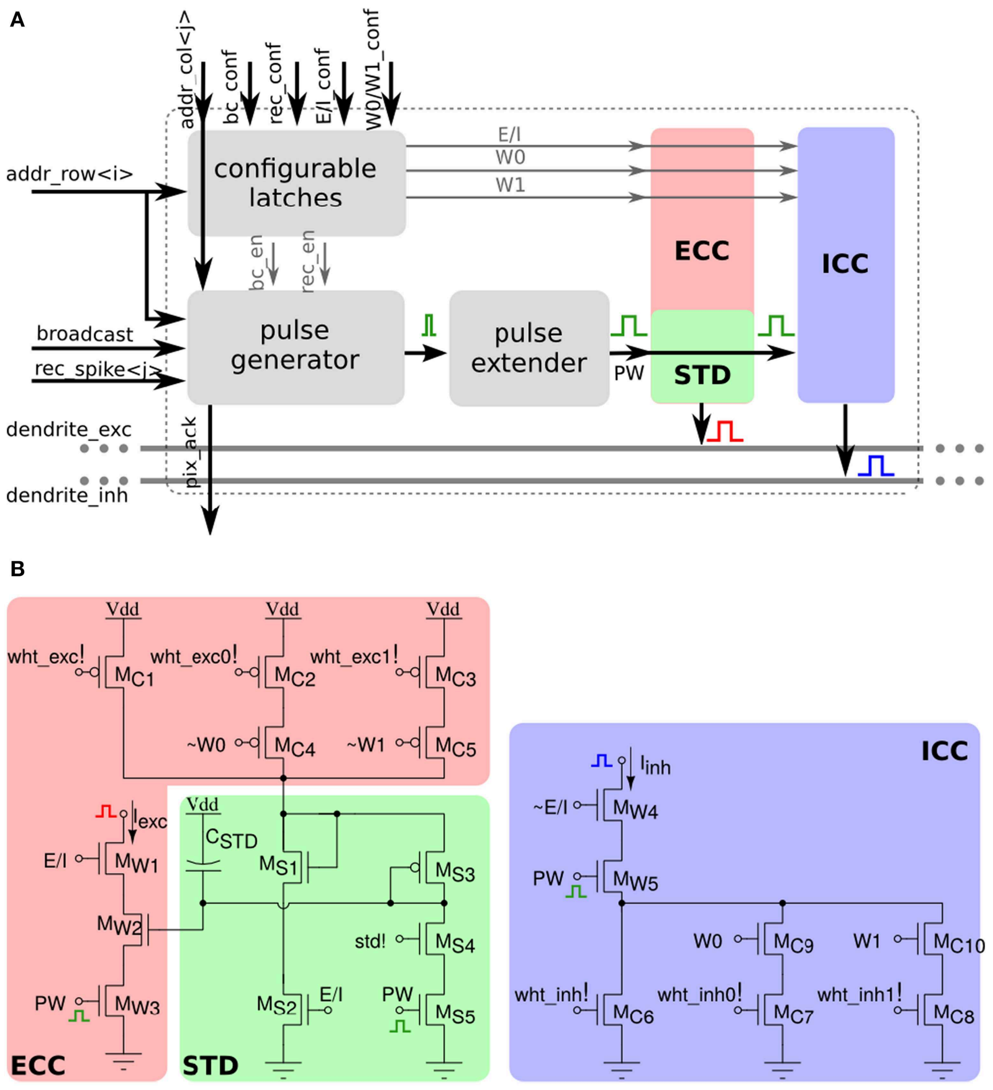


FIGURE 9 | Short-term plasticity synapse array element. (A) Block diagram of the synapse element. **(B)** Transistor level schematic diagram of the excitatory and inhibitory pulse-to-current converters.

of the addressed element. *Local configuration* patterns include commands for setting the type of synapse, programming its weight, or enabling broadcast or recurrent connections. Finally, the *Global configuration* inputs are decoded into configuration signals that represent global variables, stored onto registers in the periphery (rather than within the synapse or neuron elements). For example, the signals used to set the state of the synapse de-multiplexer are *Global configuration* signals. See the Supplementary Material for additional details on these circuits.

2.2.4.2. AER output

Each of the 256 neurons is assigned an 8-bit address for the output bus. When a neuron spikes, its address is instantaneously sent to the output AER circuits using the common four-phase handshaking scheme. Although neurons operate in a fully parallel fashion, their AEs can only access the shared output bus in a serial fashion. To manage possible simultaneous spike collisions the output AER circuits include an arbiter circuit that only grants access to the external bus to one neuron at a time. Details of these circuits are provided in the Supplementary Material.

3. Results

Here we demonstrate the capabilities of the ROLLS neuromorphic processor device with examples of hardware emulation of computational neuroscience models and pattern recognition in a machine vision task.

3.1. Attractor Networks

In this experiment we explored the collective dynamics of multiple populations of spiking silicon neurons that emulate

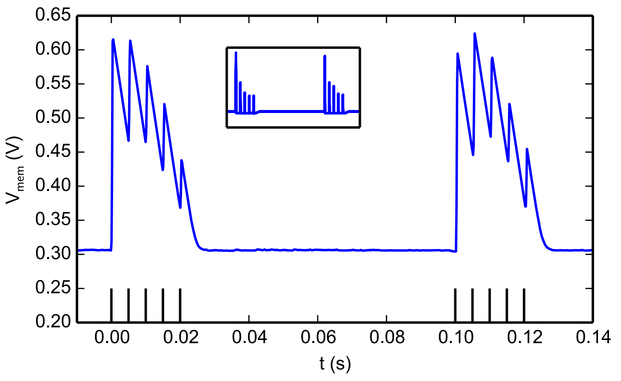


FIGURE 10 | The effect of short-term depression on EPSC magnitudes. Two bursts separated by 100 ms were sent to a programmable synapse. Each burst has 5 spikes with an inter-spike interval of 5 ms. Within a burst, The jumps in the neuron V_{mem} gradually get smaller as the synapse is depressed and the magnitude of the EPSCs it generates decreases. After the first burst, the synapse efficacy recovers as can be seen in the response to the second burst. The figure inset shows the derivative of the membrane potential which is equivalent to the synaptic EPSCs (minus the neuron leak).

the biophysics of cortical neurons organized in attractor networks (Amit, 1992). These types of networks are considered a basic computational primitive of neural processing systems. Their ability to exhibit self sustained activity is thought to be one of the basic requirements for exhibiting multiple types of cognitive processes and functions. Their collective dynamics represents the neural correlates of processes involved in working memory, perceptual decision making and attention.

We implemented the hardware attractor networks following the theories and methods proposed in Amit (1992); Wang (1999); Amit and Mongillo (2003); Del Giudice et al. (2003); Giulioni et al. (2012). We constructed an architecture comprising six pools of neurons recurrently connected. Specifically, there are three pools of 64 excitatory neurons and three pools of 10 inhibitory neurons. Neurons in each pool receive local excitation via recurrent connections implemented via the on-chip long-term synaptic plasticity circuits. In **Figure 11** each point represents a synaptic contact (i.e., an active synapse in the corresponding STP or LTP synaptic matrix). The recurrent connectivity via the LTP synapses is set to have a probability of 70% for the excitatory connections and 40% for the inhibitory ones, i.e., they have connectivity parameters $c^e_{ee} = 0.7$, $c^e_{ii} = 0.4$, respectively (see dots in **Figure 11A**). We further configured the connectivity matrix of the STP synapses such that every excitatory pools of neurons is homogeneously connected with all other excitatory pools with excitatory connectivity parameter $c^e_{ee} = 0.2$ and inhibitory connectivity parameter $c^i_{ee} = 0.2$. Inhibitory pools of neurons are connected to their corresponding excitatory pools (e.g., inhibitory pool #1 is connected to excitatory pool #1) via inhibitory synapses, with a connectivity parameter $c^i_{ei} = 0.4$. Excitatory pools of neurons are connected to their respective inhibitory pools of neurons via the STP excitatory synapses, with a connectivity parameter $c^e_{ie} = 0.7$. The behavior of the network when stimulated by a external transient stimuli is shown in **Figure 11B**. The profile of the external stimuli is depicted by the square waves below the plot of **Figure 11B**. The different colors indicate inputs to the different corresponding populations. The input stimuli are a series of Poisson spike trains, generated artificially and sent via the AER protocol to the chip virtual synapses. The mean rate of the input spike trains is $\nu_{in} = 100\,Hz$ and their duration is $t = 0.5\,s$. When the attractor networks are being driven by external stimuli their activity reaches a mean rate of approximately 50 Hz and, after the removal of these stimuli, the pools of neurons relax to a sustained state of activity of about 15 Hz, indicating that the neurons settled into their attractor states. This persistent activity is the neural correlate of working memory and can be exploited as an asynchronous distributed memory state that has peculiar dynamical properties of error correction, pattern completion, and stability against distractors (Amit, 1992).

If a population is in an attractor state, a transient stimulus to a different pool of neurons shuts down its activity via direct inhibitory connections (on the STP synaptic matrix), and brings the newly stimulated pool of neurons into a new attractor state. If we inhibit an active pool of neurons directly, with an external stimulus the population is reset and becomes inactive. This is evident in **Figure 11B** at $t = 3\,s$, when a Poisson stimulus of mean rate $\nu = 200\,Hz$ is used to inhibit all attractor networks. This experiment demonstrates how it is possible to implement robust state dependent computation and reliable memory storage using sets of 64 slow and imprecise silicon neurons. A similar, but more elaborate experiment showing how these types of circuits can be used to synthesize context-dependent behavior in neuromorphic agents, in the context of cognitive computation was recently presented in Neftci et al. (2013), using the same types of circuits and principles. The implementation of plausible neural collective dynamics in neuromorphic substrates is an important step also for future nano-technologies that are likely to be affected by device mismatch and unreliability characteristics.

3.2. Multi-Perceptron Network

Neuromorphic systems are an ideal electronic substrate for real-time, low-latency machine vision (Serrano-Gotarredona et al., 2008; Delbruck and Lang, 2013; O'Connor et al., 2013). Here we present a feasibility study which demonstrates how the ROLLS neuromorphic processor can be used in conjunction with a spiking vision sensor for learning to solve an image classification task. In this experiment (see **Figure 12**), we used a DVS, interfaced to our chip via a commercially available digital board, used to route signals from the vision sensor to the chip. We implemented a two-layer spiking neural network which processes the visual stimuli by extracting sparse random features in real-time. The network is composed of 128 VLSI hidden neurons and 128 VLSI output neurons on the ROLLS neuromorphic processor. We trained 64 of the VLSI output neurons of the network to become selective to one of two image classes, and the other 64 to become selective to the other class, via supervised learning protocol.

The experimental protocol consists of showing a sequence of static images of objects from the Caltech 101 dataset coupled with a teacher signal to steer the activity of the output neurons. The DVS is put in front of a screen where the images are

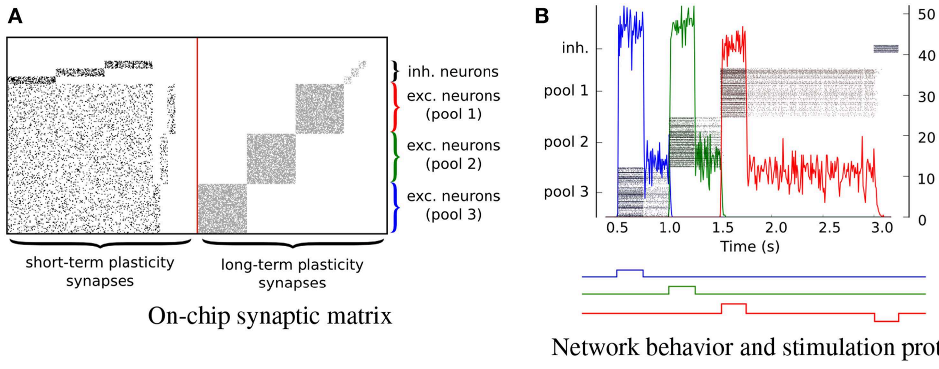


FIGURE 11 | Attractor networks. Three clustered pools of 64 neurons are configured as attractor networks. **(A)** The on-chip synaptic matrices in which every dot represents a connected synapse. **(B)** The output spiking activity of the network. Every pool of neuron is a competing working memory. The timing of the inputs is shown with square waves under the plot. Different colors represent different inputs. The down jump of the red line represents the stimulus to the inhibitory pool of neuron responsible for switching off the on-chip memory.

displayed. During the presentation, the images are flashed with a small jitter around the center of the visual field to simulate microsaccadic eye movements. The movement causes the DVS retina to continuously stream spike trains corresponding to the edges of the objects in the image. The spike trains are then routed to the STP synapse array, stimulating a population of neurons corresponding to the hidden layer of the neural network. The spikes from the hidden layer neurons are internally routed to the LTP plastic synapse array, thus activating the neurons of the output layer. With every training image, a corresponding teacher signal is provided to one of the two subgroups of the output layer neurons, depending on the image class, to associate stimulus with class. To remove artifacts generated during the transition from one presentation of an image to the next, we gated the DVS spikes, simulating a saccadic suppression mechanism analogous to the one observed in biology (Ross et al., 1996).

The performance of this experiment strongly depends on the right choice of parameters for the neural and synaptic dynamics. For this particular demonstration we chose to disable most of the complex aspects of the neural dynamics and optimized neuron and synapse parameters to obtain reasonable activity patterns in the hidden layer neurons. The activity in this layer is indeed the most important since it drives the plastic synapses that belong to the output layer neurons.

After training, our classification system was able to respond selectively to natural images of cars and motorbikes taken from the Caltech 101 database. Although an extensive characterization of the system's ability to perform object recognition is out of the scope of this work, we draw the following conclusions from our experiment:

- The choice of fixed, random projections from the input layer was surprisingly effective, though certainly not optimal for the task at hand.
- A better solution would be to include an unsupervised learning stage in the training protocol to optimize the weights of the convolution layer as in traditional machine learning approaches (LeCun et al., 1998; Le et al., 2012) and in neural systems (Olshausen and Field, 1997; Masquelier et al., 2009; Nessler et al., 2009). However, this stage would require the presentation of a large number of patterns and sophisticated synaptic plasticity rules.

Our network of randomly connected neurons projects the input stimuli into a high-dimensional space where they can be classified by linear models but with far less parameter optimization (Barak and Rigotti, 2011). This strategy is related to some of the state-of-the-art machine learning algorithms for pattern classifications, such as Support Vector Machines (SVMs) (Vapnik, 1995). Clearly, the generalization properties of our system are not comparable to standard machine learning approaches but they are also expected to scale with the number of randomly connected neurons in the hidden layer (Rigotti et al., 2010; Barak et al., 2013). Notice also that we haven't exploited any temporal structure of the input data, though we recently demonstrated that our hardware supports this functionality (Sheik et al., 2012a,b, 2013). For cases in which the temporal structure of the input stimuli is relevant, it would be possible to follow alternative approaches, for example by interconnecting the neurons in the hidden layer to form a Liquid State Machine (LSM) (Maass et al., 2002). This solution would be particularly interesting in situations where information hidden in the fine temporal structure is expected to impact the performance of the recognition system. Also for this approach, it would be sufficient to provide an output layer analogous to the one used in our experiment, that could be trained in an analogous way. In our example we used multiple neurons clustered into two distinct pools in the output layer for our simple two-class discrimination problem, (e.g., instead of using just two output neuron units). The rationale behind this choice is that, given the many sources of noise in the system (the micro-saccadic movements, the DVS spiking output, the stochastic plasticity mechanism, the hardware mismatch), each neuron taken singularly is not expected to perform well on the task (i.e., it will implement a "weak" classifier, showing low class specificity). However, the performance of the

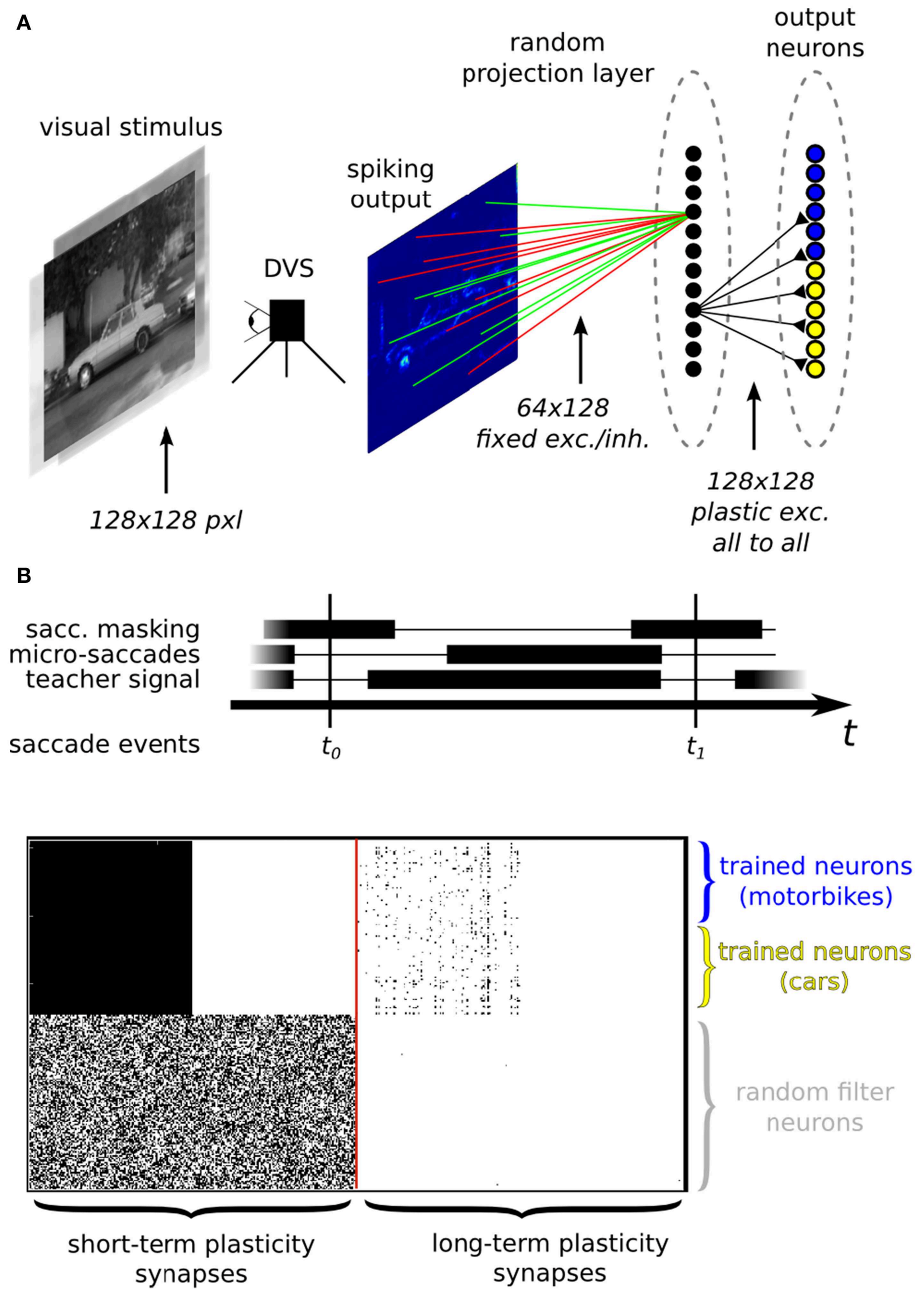


FIGURE 12 | (A) Image classification example using inputs from a DVS. **(A)** Top: neural network architecture. Two different classes of images (here motorbikes or cars) are displayed on a screen with a small jitter applied at 10 Hz. A random subset of the spikes emitted by the DVS are mapped to 128 hidden layer neurons. Specifically, each of the 128 neurons is connected to 64 randomly selected pixels with either positive or negative weights, also set at random. The output neurons in the last layer receive spikes from all the 128 hidden layer neurons, via plastic synapses. The output layer neurons are also driven by an external "teacher" signal which is correlated with one of the image classes. **(A)** Bottom: diagram of the experimental protocol timeline. Notice the presence of a saccade inhibition mechanism which electronically suppresses DVS input during a virtual saccade, i.e., when the displayed image is replaced with the next one. **(B)** Synaptic matrices of the ROLLS neuromorphic processor showing the hardware configuration of the classification neural network. The STP synapses represent the synapses of the hidden layer; the LTP synapses represent the synapses of the output layer.

overall system improves as responses aggregated from multiple neurons are considered. This can be visually appreciated from the raster plots of **Figure 13** where only population-level firing rates are selective for the input classes, but not the single neuron activities. This phenomenon is directly related to a notorious machine learning technique that uses "boosting" to improve the

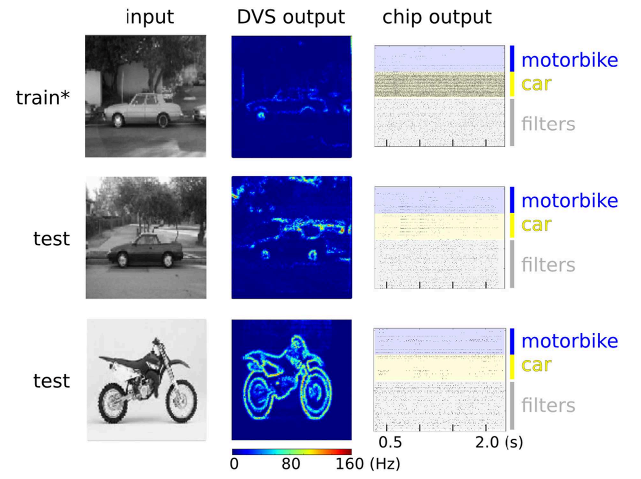


FIGURE 13 | Spiking activity of the hardware neurons during the training (upper panel) and testing phase (middle and lower panels). **Left column**: examples of raw images from the Caltech 101 database. **Middle column**: heat map of the DVS spiking activity, where each pixel color represents the pixel's mean firing rate. **Right column**: raster plots of the ROLLS neuromorphic processor neurons. The star on the top panel label indicates that during the training phase an additional excitatory "teacher" signal was used to stimulate the "car" output neurons to induce plasticity. During testing no teacher signal is provided and only the excitatory currents from the input synapses drive the classifier activities. The average firing rates of the output layer for "motorbike" and "car" neuron pools are 6.0 Hz and 80.9 Hz during training. During testing with a "car" image they are 7.1 Hz and 11.1 Hz. During testing with a "motorbike" image they are 7.4 Hz and 4.9 Hz.

performance of weak-classifiers (Breiman, 2001; Schapire and Freund, 2012).

4. Discussion

Unlike conventional von Neumann processors that carry out bit-precise processing and access and store data in a physically separate memory block, the ROLLS neuromorphic processor uses elements in which memory and computation are co-localized. The computing paradigm implemented by these types of neuromorphic processors does not allow for the virtualization of time, with the transfer of partial results back and forth between the computing units and physically separate memory banks at high speeds. Instead, their synapse and neuron circuits process input spikes on demand as they arrive, and produce their output responses in real-time. Consequently, the time constants of the synapses and neurons present in these devices need to be well-matched to the signals the system is designed to process. For the case of real-time behaving systems that must interact with the environment, while processing natural signals in real-time, these time constants turn out to be compatible with the biologically plausible ones that we designed into the ROLLS neuromorphic processor. As we implemented non-linear operations in each synapse (such as short-term depression or long-term plasticity), it is not possible to time-multiplex linear circuits to reduce the area occupied by the synaptic matrix array. As a consequence, our device is essentially a large memory chip with dedicated circuits for each synapse that act both as memory elements and computing ones. This approach is complementary to other recent ones that focus on accelerated neural simulations (Bruederle et al., 2011), or that target the real-time emulation of large populations of neurons but with no on-chip learning or adaptive behaviors at the synapse level (Benjamin et al., 2014).

The device we describe here is ideal for processing sensory signals produced by neuromorphic sensors (Liu and Delbruck, 2010) and building autonomous behaving agents. The system level examples demonstrated in Section 3 show how this can be achieved in practice: the hardware attractor network experiment focuses on the idea that the functional units of the cortex are subset of neurons that are repeatedly active together and shows that such units have the capability of storing state-dependent information; the pattern classification experiment demonstrates how it is possible to implement relatively complex sensory processing tasks using event-based neuromorphic sensors.

Our results demonstrate the high-degree of programmability of our device as well as its usability in typical application domains. Its properties make it an ideal tool for exploring computational principles of spiking systems consisting of both spiking sensors and cortical-like processing units. This type of tools are an essential resource for understanding how to leverage the physical properties of the electronic substrate as well as the most robust theories of neural computation in light of the design of a new generation of cortex-like processors for real-world applications. The multi-chip system is supported by the use of a newly developed software front-end, PyNCS, which allows rapid integration of heterogeneous spiking neuromorphic devices in unique hardware infrastructure and continuous online monitoring and interaction with the system during execution (Stefanini et al., 2014). In order to integrate the DVS and ROLLS in the existing software and hardware infrastructure, it was necessary to list the address specifications for the spiking events and for the configuration events in Neuromorphic Hardware Mark-up Language (NHML) files, the neuromorphic mark-up language used by PyNCS to control the neuromorphic system.

The potential of the approach proposed in this work for building intelligent autonomous systems is extremely high, as we develop brain-inspired computing devices embedded with learning capabilities that can interact with the environment in real time. Substantial progress has already been made in the theoretical domain (Schöner, 2007; Rutishauser and Douglas, 2009), and preliminary results have already been demonstrated also with neuromorphic cognitive systems (Neftci et al., 2013) synthesized by the user. The ROLLS neuromorphic processor described in this work can therefore contribute to extending the current state-of-the-art by providing also adaptation and learning mechanisms that could allow these systems to learn the appropriate network properties to implement autonomous cognitive systems.

5. Conclusions

We presented a mixed-signal analog/digital VLSI device for implementing on-line learning spiking neural network architectures with biophysically realistic neuromorphic circuits such as STP synapses, LTP synapses and low-power, low-mismatch adaptive I&F silicon neurons. The proposed architecture exploits digital configuration latches in each synapse and neuron element to guarantee a highly flexible infrastructure for programming, with the same device, diverse spiking neural network architectures.

All the operations of the chip are achieved via asynchronous AE streams. These operations include sending events to the chip, configuring the topology of the neuron network, probing internal variables, as well as programming internal properties of synapse and neurons. The parameters for different synapse and neuron behaviors can be fine tuned by programming the temperature-compensated on-chip BG.

The ROLLS neuromorphic processor can be used to carry out basic research in computational neuroscience and can be exploited for developing application solutions for practical tasks. In particular, this architecture has been developed to study spike-based adaptation and plasticity mechanism and to use its ability to carry out on-chip on-line learning for solving tasks that require the system to adapt to the changes in its input signals and in the environment it interacts with.

Funding

This work was funded by the EU ERC Grant neuroP (257219), the EU FET Grant SI-CODE (284553) and the Samsung Advanced Institute of Technology (SAIT)'s Global Research Outreach (GRO) Program.

Acknowledgments

Some of the circuits described in this work were developed jointly with Chiara Bartolozzi, Elisabetta Chicca, and Srinjoy Mitra. We thank Stefano Fusi, Mattia Rigotti, Paolo Del Giudice, and Massimiliano Giulioni for fruitful discussions on the learning mechanisms and silicon neuron attractor networks; we thank Adrian Whatley and Sadique Sheik for their helpful comments on the overall chip architecture, Hongzhi You for his work on minimizing mismatch in the silicon neuron circuit, Giovanni Rovere for designing the rail-to-rail voltage buffer used to read out the analog signals, and Tobi Delbruck for technical support on designing the on-chip Bias Generator. We also acknowledge all our other colleagues at the Institute of Neuroinformatics for fruitful discussions and collaborations.

Supplementary Material

The Supplementary Material for this article can be found online at: http://journal.frontiersin.org/article/10.3389/fnins.2015.00141/abstract

References

Abbott, L., and Nelson, S. (2000). Synaptic plasticity: taming the beast. *Nat. Neurosci.* 3, 1178–1183. doi: 10.1038/81453

Amit, D. (1992). *Modeling Brain Function: The World of Attractor Neural Networks*. New York, NY: Cambridge University Press.

Amit, D., and Mongillo, G. (2003). Spike-driven synaptic dynamics generating working memory states. *Neural Comput.* 15, 565–596. doi: 10.1162/089976603321192086

Barak, O., and Rigotti, M. (2011). A simple derivation of a bound on the perceptron margin using singular value decomposition. *Neural Comput.* 23, 1935–1943. doi: 10.1162/NECO_a_00152

Barak, O., Rigotti, M., and Fusi, S. (2013). The sparseness of mixed selectivity neurons controls the generalization–discrimination trade-off. *J. Neurosci.* 33, 3844–3856. doi: 10.1523/JNEUROSCI.2753-12.2013

Bartolozzi, C., and Indiveri, G. (2007a). "A selective attention multi–chip system with dynamic synapses and spiking neurons," in *Advances in Neural Information Processing Systems (NIPS), Neural Information Processing Systems Foundation*, Vol. 19, eds B. Schölkopf, J. Platt, and T. Hofmann (Cambridge, MA: MIT Press), 19, 113–120.

Bartolozzi, C., and Indiveri, G. (2007b). Synaptic dynamics in analog VLSI. *Neural Comput.* 19, 2581–2603. doi: 10.1162/neco.2007.19.10.2581

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J., et al. (2014). Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 699–716. doi: 10.1109/JPROC.2014.2313565

Billings, G., and van Rossum, M. (2009). Memory retention and spike-timing-dependent plasticity. *J. Neurophysiol.* 101, 2775–2788. doi: 10.1152/jn.91007.2008

Boahen, K. (2000). Point-to-point connectivity between neuromorphic chips using address-events. *IEEE Trans. Circ. Syst. II* 47, 416–434. doi: 10.1109/82.842110

Boegerhausen, M., Suter, P., and Liu, S.-C. (2003). Modeling short-term synaptic depression in silicon. *Neural Comput.* 15, 331–348. doi: 10.1162/089976603762552942

Brader, J., Senn, W., and Fusi, S. (2007). Learning real world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Breiman, L. (2001). Random forests. *Mach. Learn.* 45, 5–32. doi: 10.1023/A:1010933404324

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Bruederle, D., Petrovici, M., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., et al. (2011). A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol. Cybern*. 104, 263–296. doi: 10.1007/s00422-011-0435-9

Chicca, E., Stefanini, F., Bartolozzi, C., and Indiveri, G. (2014). Neuromorphic electronic circuits for building autonomous cognitive systems. *Proc. IEEE* 102, 1367–1388. doi: 10.1109/JPROC.2014.2313954

Costas-Santos, J., Serrano-Gotarredona, T., Serrano-Gotarredona, R., and Linares-Barranco, B. (2007). A spatial contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems. *IEEE Trans. Circ. Syst. I*, 54, 1444–1458. doi: 10.1109/TCSI.2007.900179

Del Giudice, P., Fusi, S., and Mattia, M. (2003). Modeling the formation of working memory with networks of integrate-and-fire neurons connected by plastic synapses. *J. Physiol. Paris* 97, 659–681. doi: 10.1016/j.jphysparis.2004.01.021

Deiss, S., Douglas, R., and Whatley, A. (1998), "A pulse-coded communications infrastructure for neuromorphic systems," in *Pulsed Neural Networks chapter 6* eds W. Maass and C. Bishop (Cambridge: MIT Press), 157–178.

Delbruck, T., Berner, R., Lichtsteiner, P., and Dualibe, C. (2010). "32-bit configurable bias current generator with sub-off-current capability," in *IEEE International Symposium on Circuits and Systems (ISCAS)* (Paris: IEEE), 1647–1650.

Delbruck, T., and Lang, M. (2013). Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Front. Neurosci.* 7:223. doi: 10.3389/fnins.2013.00223

Douglas, R., Mahowald, M., and Mead, C. (1995). Neuromorphic analogue VLSI. *Annu. Rev. Neurosci.* 18, 255–281.

Farabet, C., Couprie, C., Najman, L., and Le Cun, Y. (2013). Learning hierarchical features for scene labeling, *IEEE Trans. Patt. Anal. Mach. Intell.* 35, 1915–1929. doi: 10.1109/TPAMI.2012.231

Farquhar, E., and Hasler, P. (2005). A bio-physically inspired silicon neuron. *IEEE Trans. Circ. Syst.* 52, 477–488. doi: 10.1109/TCSI.2004.842871

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

Giulioni, M., Camilleri, P., Dante, V., Badoni, D., Indiveri, G., Braun, J., et al. (2008). "A VLSI network of spiking neurons with plastic fully configurable "stop-learning" synapses," in *International Conference on Electronics, Circuits, and Systems, ICECS* (IEEE), 678–681.

Giulioni, M., Camilleri, P., Mattia, M., Dante, V., Braun, J., and Giudice, P. D. (2012). Robust working memory in an asynchronously spiking neural network realized in neuromorphic VLSI. *Front. Neurosci.* 5:149. doi: 10.3389/fnins.2011.00149

Hsieh, H.-Y., and Tang, K.-T. (2012). Vlsi implementation of a bio-inspired olfactory spiking neural network. *IEEE Trans. Neural Netw. Learn. Syst.* 23, 1065–1073. doi: 10.1109/TNNLS.2012.2195329

Hynna, K., and Boahen, K. (2007). Thermodynamically-equivalent silicon models of ion channels. *Neural Comput.* 19, 327–350. doi: 10.1162/neco.2007.19.2.327

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike–timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Indiveri, G., Linares-Barranco, B., Hamilton, T., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Izhikevich, E. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., et al. (2012). "Building high-level features using large scale unsupervised learning," in *International Conference in Machine Learning* (Edinburgh).

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324.

Linares-Barranco, B., and Serrano-Gotarredona, T. (2003). On the design and characterization of femtoampere current-mode circuits. *IEEE J. Solid State Circ.* 38, 1353–1363. doi: 10.1109/JSSC.2003.814415

Lisman, J., and Spruston, N. (2010). Questions about stdp as a general model of synaptic plasticity. *Front. Synaptic Neurosci.* 2:1–3. doi: 10.3389/fnsyn.2010.00140

Liu, S.-C. (2003). Analog VLSI circuits for short-term dynamic synapses. *Eur. J. Appl. Signal Process.* 7, 1–9. doi: 10.1155/S1110865703302094

Liu, S.-C., and Delbruck, T. (2010). Neuromorphic sensory systems. *Curr. Opin. Neurobiol.* 20, 288–295. doi: 10.1016/j.conb.2010.03.007

Liu, S.-C., Kramer, J., Indiveri, G., Delbruck, T., and Douglas, R. (2002). *Analog VLSI: Circuits and Principles*. Cambridge; London: MIT Press.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* 14, 2531–2560. doi: 10.1162/089976602760407955

Mahowald, M., and Douglas, R. (1991). A silicon neuron. *Nature* 354, 515–518.

Manohar, R. (2006). "Reconfigurable asynchronous logic," in *Custom Integrated Circuits Conference* (San Jose, CA: IEEE), 13–20.

Markram, H., Gerstner, W., and Sjöström, P. (2012). Spike-timing-dependent plasticity: a comprehensive overview. *Front. Synaptic Neurosci.* 4:2. doi: 10.3389/fnsyn.2012.00002

Masquelier, T., Guyonneau, R., and Thorpe, S. (2009). Competitive STDP-based spike pattern learning. *Neural Comput.* 21, 1259–1276. doi: 10.1162/neco.2008.06-08-804

Merolla, P., Arthur, J., Alvarez-Icaza, R., Cassidy, A., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Mohamed, A., Dahl, G., and Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Trans. Audio Speech Lang. Process.* 20, 14–22. doi: 10.1109/TASL.2011.2109382

Neftci, E., Binas, J., Rutishauser, U., Chicca, E., Indiveri, G., and Douglas, R. (2013). Synthesizing cognition in neuromorphic electronic systems. *Proc. Natl. Acad. Sci. U.S.A.* 110, E3468–E3476. doi: 10.1073/pnas.1212083110

Neil, D., and Liu, S.-C. (2014). Minitaur, an event-driven FPGA-based spiking network accelerator. *IEEE Trans. Very Large Scale Integr. Syst.* 99, 1–1. doi: 10.1109/TVLSI.2013.2294916

Nessler, B., Pfeiffer, M., and Maass, W. (2009). "STDP enables spiking neurons to detect hidden causes of their inputs," in *Advances in Neural Information Processing Systems (NIPS)*, Vol. 22, eds Y. Bengio, D. Schuurmans, J. Lafferty, C. I. Williams, and A. Culotta (Vancouver, BC: MIT Press), 1357–1365.

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* 7:178. doi: 10.3389/fnins.2013.00178

Olshausen, B., and Field, D. (1997). Sparse coding with an overcomplete basis set: a strategy employed by vi? *Vis. Res.* 37, 3311–3326.

Petrovici, M., Vogginger, B., Müller, P., Breitwieser, O., Lundqvist, M., Muller, L., et al. (2014). Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PLoS ONE* 9:e108590. doi: 10.1371/journal.pone.0108590

Ramakrishnan, S., Wunderlich, R., and Hasler, P. (2012). "Neuron array with plastic synapses and programmable dendrites," in *Biomedical Circuits and Systems Conference (BioCAS)* (Hsinchu: IEEE), 400–403.

Rasche, C., and Hahnloser, R. (2001). Silicon synaptic depression. *Biol. Cybern.* 84, 57–62. doi: 10.1007/s004220170004

Rigotti, M., Rubin, D. B. D., Morrison, S., Salzman, C., and Fusi, S. (2010). Attractor concretion as a mechanism for the formation of context representations. *Neuroimage* 52, 833–847. doi: 10.1016/j.neuroimage.2010.01.047

Ross, J., Burr, D., and Morrone, C. (1996). Suppression of the magnocellular pathway during saccades. *Behav. Brain Res.* 80, 1–8.

Rovere, G., Ning, Q., Bartolozzi, C., and Indiveri, G. (2014). "Ultra low leakage synaptic scaling circuits for implementing homeostatic plasticity in neuromorphic architectures," in *International Symposium on Circuits and Systems, (ISCAS)* (Melbourne VIC: IEEE), 2073–2076.

Rutishauser, U., and Douglas, R. (2009). State-dependent computation using coupled recurrent networks. *Neural Comput.* 21, 478–509. doi: 10.1162/neco.2008.03-08-734

Sarpeshkar, R., Lyon, R., and Mead, C. (1996). "An analog VLSI cochlea with new transconductance amplifiers and nonlinear gain control," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Vol. 3 (Atlanta, GA: IEEE), 292–296.

Schapire, R., and Freund, Y. (2012). *Boosting: Foundations and Algorithms.* Cambridge, MA: MIT Press.

Schmuker, M., Pfeil, T., and Nawrot, M. (2014). A neuromorphic network for generic multivariate data classification. *Proc. Natl. Acad. Sci.* 111, 2081–2086. doi: 10.1073/pnas.1303053111

Schöner, G. (2007). *Cambridge Handbook of Computational Cognitive Modeling, Chapter Dynamical Systems Approaches to Cognition.* New York, NY: Cambridge University Press.

Senn, W. (2002). Beyond spike timing: the role of nonlinear plasticity and unreliable synapses. *Biol. Cybern.* 87, 344–355. doi: 10.1007/s00422-002-0350-1

Senn, W., and Fusi, S. (2005). Learning only when necessary: better memories of correlated patterns in networks with bounded synapses. *Neural Comput.* 17, 2106–2138. doi: 10.1162/0899766054615644

Serrano-Gotarredona, R., Serrano-Gotarredona, T., Acosta-Jimenez, A., Serrano-Gotarredona, C., Perez-Carrasco, J., Linares-Barranco, A., et al. (2008). On real-time aer 2d convolutions hardware for neuromorphic spike based cortical processing. *IEEE Trans. Neural Netw.* 19, 1196–1219. doi: 10.1109/TNN.2008.2000163

Sheik, S., Chicca, E., and Indiveri, G. (2012a). "Exploiting device mismatch in neuromorphic VLSI systems to implement axonal delays," in *International Joint Conference on Neural Networks, IJCNN* (Brisbane, QLD: IEEE), 1940–1945.

Sheik, S., Coath, M., Indiveri, G., Denham, S., Wennekers, T., and Chicca, E. (2012b). Emergent auditory feature tuning in a real-time neuromorphic VLSI system. *Front. Neurosci.* 6:17. doi: 10.3389/fnins.2012.00017

Sheik, S., Pfeiffer, M., Stefanini, F., and Indiveri, G. (2013). "Spatio-temporal spike pattern classification in neuromorphic systems," in *Biomimetic and Biohybrid Systems,* eds N. F. Lepora, A. Mura, H. G. Krapp, P. F. M. J. Verschure, and T. J. Prescott (Zurich: Springer), 262–273.

Stefanini, F., Sheik, S., Neftci, E., and Indiveri, G. (2014). Pyncs: a microkernel for high-level definition and configuration of neuromorphic electronic systems. *Front. Neuroinform.* 8:73. doi: 10.3389/fninf.2014.00073

van Schaik, A., Jin, C., and Hamilton, T. (2010). "A log-domain implementation of the Izhikevich neuron model," in *International Symposium on Circuits and Systems, (ISCAS)* (Paris: IEEE), 4253–4256.

van Schaik, A., and Meddis, R. (1999). Analog very large-scale integrated (VLSI) implementation of a model of amplitude-modulation sensitivity in the auditory brainstem. *J. Acoust. Soc. Am.* 105, 811.

Vapnik, V. (1995). *The Nature of Statical Learning Theory*. New York, NY: Springer-Verlag.

Wang, R., Cohen, G., Stiefel, K., Hamilton, T., Tapson, J., and van Schaik, A. (2013). An FPGA implementation of a polychronous spiking neural network with delay adaptation. *Front. Neurosci.* 7:14. doi: 10.3389/fnins.2013.00014

Wang, X. (1999). Synaptic basis of cortical persistent activity: the importance of NMDA receptors to working memory. *J. Neurosci.* 19, 9587–9603.

Xu, P., Horiuchi, T.-K., Sarje, A., and Abshire, P. (2007). "Stochastic synapse with short-term depression for silicon neurons," in *Biomedical Circuits and Systems Conference (BIOCAS)* (Montreal, QC: IEEE), 99–102.

Yu, T., Park, J., Joshi, S., Maier, C., and Cauwenberghs, G. (2012). "65k-neuron integrate-and-fire array transceiver with address-event reconfigurable synaptic routing," in *Biomedical Circuits and Systems Conference (BioCAS)* (Hsinchu: IEEE), 21–24.

Zaghloul, K., and Boahen, K. (2004). Optic nerve signals in a neuromorphic chip: Parts 1&2. *IEEE Trans. Biomed. Circ. Syst.* 51, 657–675. doi: 10.1109/TBME.2003.821039

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH ARTICLE
published: 02 February 2015
doi: 10.3389/fnins.2015.00010

Switched-capacitor realization of presynaptic short-term-plasticity and stop-learning synapses in 28 nm CMOS

Marko Noack[1]*, Johannes Partzsch[1], Christian G. Mayr[2], Stefan Hänzsche[1], Stefan Scholze[1], Sebastian Höppner[1], Georg Ellguth[1] and Rene Schüffny[1]

[1] *Chair of Highly-Parallel VLSI-Systems and Neuromorphic Circuits, Technische Universität Dresden, Dresden, Germany*
[2] *Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland*

Edited by:
Themis Prodromakis, University of Southampton, UK

Reviewed by:
Theodore Yu, Texas Instruments Inc., USA
Siddharth Joshi, University of California, San Diego, USA

****Correspondence:***
Marko Noack, Chair of Highly-Parallel VLSI-Systems and Neuromorphic Circuits, Institute of Circuits and Systems, Faculty of Electrical Engineering and Information Technology, Technische Universität Dresden, 01062 Dresden, Germany
e-mail: marko.noack@tu-dresden.de

Synaptic dynamics, such as long- and short-term plasticity, play an important role in the complexity and biological realism achievable when running neural networks on a neuromorphic IC. For example, they endow the IC with an ability to adapt and learn from its environment. In order to achieve the millisecond to second time constants required for these synaptic dynamics, analog subthreshold circuits are usually employed. However, due to process variation and leakage problems, it is almost impossible to port these types of circuits to modern sub-100nm technologies. In contrast, we present a neuromorphic system in a 28 nm CMOS process that employs switched capacitor (SC) circuits to implement 128 short term plasticity presynapses as well as 8192 stop-learning synapses. The neuromorphic system consumes an area of 0.36 mm^2 and runs at a power consumption of 1.9 mW. The circuit makes use of a technique for minimizing leakage effects allowing for real-time operation with time constants up to several seconds. Since we rely on SC techniques for all calculations, the system is composed of only generic mixed-signal building blocks. These generic building blocks make the system easy to port between technologies and the large digital circuit part inherent in an SC system benefits fully from technology scaling.

Keywords: switched-capacitor neuromorphic, stop-learning synapse, dynamic synapse, deep-submicron neuromorphic, low-leakage switched-capacitor circuits

1. INTRODUCTION

Biological synapses employ a range of plasticity mechanisms in modulating their stimulus transmission. For example short-term plasticity on the timescale of hundreds of milliseconds has been identified as a crucial constituent of dynamic neural information processing, allowing for temporal filtering (Grande and Spain, 2005), selective information transmission (Mayr et al., 2009) and pattern classification in attractor networks (Mejias and Torres, 2009). Long-term plasticity, with induction on the minute to hour scale, is used for pattern learning (Brader et al., 2007) and topology formation, allowing a network to be structured for solving a particular problem (Rubinov et al., 2011). Both of these mechanisms employ exponential time windows with time constants on the order of 10–1000 ms.

Most analog neuromorphic implementations of plasticity rely on subthreshold circuits (Indiveri et al., 2006) to achieve the small currents necessary for these long time constants. However, these are hard to port to advanced CMOS techologies, since leakage currents rapidly increase with down-scaling, reaching the range of the desired signal currents (Roy et al., 2003). Some plasticity circuits have also been implemented in OTA-C architectures (Koickal et al., 2007; Noack et al., 2011), but these suffer from the same problems with small currents. Digital plasticity circuits (Cassidy et al., 2011) are not subject to this limitation, but have limited biological veracity due to their digital state variables. For subthreshold circuits, an additional problem is the increase of device mismatch and process variation (Kinget, 2005), making transistors almost unusable for the exponential computation that subthreshold circuits rely upon. This is why even recent subthreshold neuromorphic systems have been manufactured in quite large technologies (Bartolozzi and Indiveri, 2007; Indiveri et al., 2010; Moradi and Indiveri, 2013), with the sole exception a recent design in 90 nm (Park et al., 2014).

The SC technique offers a viable alternative, as it utilizes robust charge-based signal transmission. That is, it computes with charges that are equivalent to accumulating the continuous signal currents of subthreshold circuits across time, thereby raising signal levels compared to the subthreshold approach. This approach has already been successfully applied to neuromorphic neuron implementations (Vogelstein et al., 2007; Folowosele et al., 2009).

In Mayr et al. (in press), a neuromorphic system using SC circuits has been presented that achieves biological real time operation in a 28 nm CMOS process. While (Mayr et al., in press) presents the static neuromorphic components (weight implementation, neurons, etc.) and the overall system integration, in this companion paper we focus on neuronal dynamics. Specifically, this paper presents the SC circuits that implement presynaptic adaptation and synaptic plasticity. The short-term (presynaptic) plasticity has been adapted for SC (Noack et al., 2012) from the biology-derived neurotransmitter release model of Markram et al. (1998). The long-term (synaptic) plasticity circuit implements the stop learning stochastic synapse model of Brader et al. (2007). To the best of our knowledge, this represents the first time the well-known stop-learning paradigm has been translated to SC circuits.

Vogelstein et al. (2007) and Folowosele et al. (2009) have chosen a straightforward SC approach with conventional CMOS switches, as leakage currents were not a concern in their chosen technology nodes. However, this approach is not possible in deep-submicron technologies such as the employed 28 nm process. The leakage for open switches would preclude storing a signal on the required 10–1000 ms timescale. Thus, we describe circuit techniques to reduce leakage currents, in turn allowing us to achieve high time constants. The entire neuromorphic system consists of standard analog building blocks and synthesizable digital logic, making it easy to port between technologies. As detailed later, the system architecture has been optimized for mismatch reduction.

2. MATERIALS AND METHODS

2.1. OVERALL SYSTEM

Figure 1 gives an overview of the system (Mayr et al., in press). 128 input circuits at the left side realize presynaptic short-term dynamics for their respective row in the synaptic matrix (Noack et al., 2012), while the 64 neurons at the bottom are driven by their respective column, providing the output (i.e., stimulation) signal as a function of the 8192 synapses in the system, which couple presynaptic input to neurons. Synaptic weights are stored in a dedicated RAM block separate from the synapse matrix.

The entire driving circuitry of presynapses, synapses and neurons is situated at the left hand side of the matrix. A state machine cycles through the columns of the synaptic matrix. At the start of the cycle, the input pulses that were registered during the last cycle are forwarded to the driver circuits and the corresponding presynaptic adaptation state is computed. Then, each synaptic column is activated sequentially, and the synaptic plasticity change of a synapse at a specific row is computed based on presynaptic pulse activity of that row and the membrane state of the neuron of the current column. Concurrently, the presynaptic pulses are integrated on the neuron. Sharing the active driver circuitry for all neurons respectively for all synapses of a row inherently reduces mismatch effects, as the only remaining mismatch between synapses is the mismatch of their state-holding capacitors. Mismatch between transistors, i.e., between active circuits, is only felt between rows.

The circuit design utilizes only digital core devices of the 28 nm SLP (super low power) technology. In contrast to the current biasing usually employed in neuromorphic ICs (Yang et al., 2012), the neuromorphic SC circuits require voltages provided by a digital-to-analog converter (DAC) to set amplitude parameters such as scaling of presynaptic adaptation, etc. This saves pins and offers an easy and robust configurability.

Time constants are set via counters that govern the switching cycles of the SC circuits. Thus, scaling of the clock frequency effectively scales the speed of the system, keeping the resolution

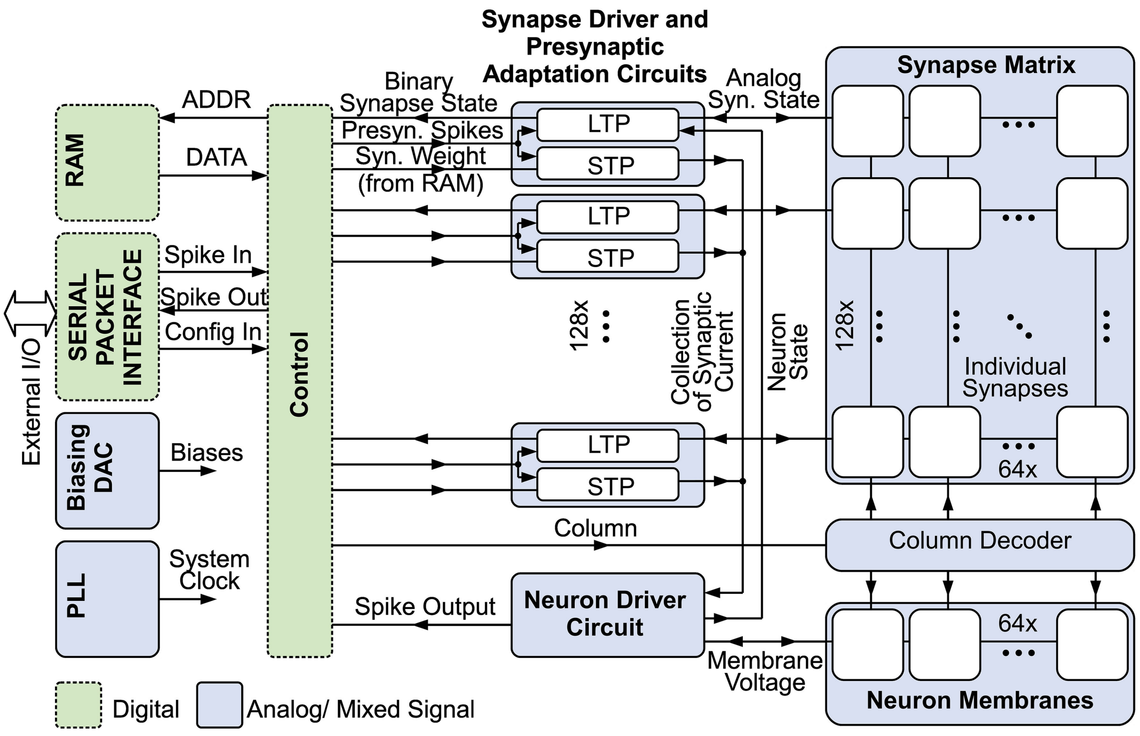


FIGURE 1 | Overview of the neuromorphic system with mixed signal SC blocks (e.g., presynaptic adaptation, synapse matrix and neurons), digital control, synaptic weight RAM, biasing DAC, PLL clock input and serial packet IO (Mayr et al., in press).

relative to the chosen time base. As the clock speed scaling retains the relative speed of all processes, the same configuration for all parameters (amplitudes and time constants) can be used irrespective of the speed-up, nominally giving the same results. The neuromorphic system was designed for speeds from biological real-time (corresponding to a 0.62 ms full cycle time of the synaptic matrix) up to an acceleration of 100.

Communication with the system is provided by a JTAG interface, implementing a generic packet-based protocol. Similar to the communication setup in Hartmann et al. (2010); Scholze et al. (2011), these packets contain configuration and incoming/outgoing pulse communication data. Additionally, two configurable test outputs allow for monitoring analog voltages, such as membrane potentials. With its minimal interface, using only 6 signal pins and two bias pins (one bias current and one pin for common mode voltage), the neuromorphic system can be easily integrated into a multi-core system mediated by an FPGA. A chip photograph is shown in **Figure 2**. The neuromorphic system occupies 0.36 mm^2 and is surrounded by various test structures. The overall IC has a size of 1.5 mm × 3 mm.

2.2. IMPLEMENTATION OF PRESYNAPTIC SHORT-TERM PLASTICITY

2.2.1. Model

The presynaptic adaptation circuit implements the model of synaptic dynamics proposed in Noack et al. (2012), which is derived from a model based on biological measurements (Markram et al., 1998). The major drawback of the original approach in Markram et al. (1998) with respect to a switched-capacitor implementation is the need for a wide-range voltage multiplier for calculating the product of the facilitation and depression state variables. Existing multipliers are rather complex, very area consuming (Hong and Melchior, 1984) or need large operational amplifiers driving resistive loads (Khachab and Ismail, 1991). In contrast, the model proposed in Noack et al. (2012) is capable of approximately reproducing the original model without any multiplier circuit and with a minimum effort on analog circuitry in general.

The iterative description of the proposed model is shown in Equations (1–3):

$$u_{n+1} = u_n \cdot (1 - U) \cdot \mathrm{e}^{-\frac{\Delta t_n}{\tau_u}} + U \tag{1}$$

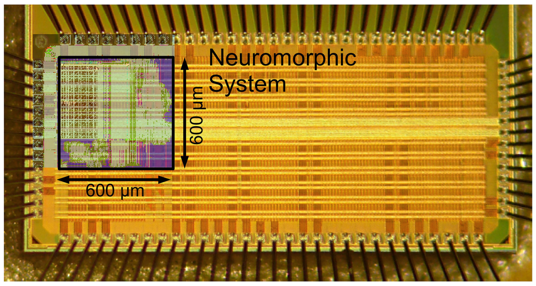


FIGURE 2 | Chip photograph with overlay of the 600 µm × 600 µm neuromorphic system layout. Die size is 1.5 mm × 3 mm (Mayr et al., in press).

$$R_{n+1} = ((1 - \alpha) \cdot R_n + \alpha \cdot u_n) \cdot \mathrm{e}^{-\frac{\Delta t_n}{\tau_R}} \tag{2}$$

$$PSC_n = A \cdot (u_n - R_n). \tag{3}$$

It provides the amplitude PSC_n of the postsynaptic current for successive presynaptic spikes incorporating their spiking history, where n is the number of the observed spike and Δt_n denotes the time between n-th and $(n + 1)$-th spike. The model is capable of reproducing facilitation and depression as well as various combinations of both mechanisms. Facilitation is modeled by variable u, which is adopted from Markram et al. (1998). At each incoming presynaptic spike u is increased by a certain amount, depending on U. Between spikes it exponentially decays back to U with time constant τ_u. Thus, u is bound to the interval $[U, 1]$. Variable R describes the depression mechanism and is also increased at every presynaptic spike. Inspired from Markram et al. (1998) the amount depends on the current value of u. The strength of depression is controlled via α, which can be any value between 0 and 1. Between spikes R decays back to 0 with time constant τ_R. The resulting PSC amplitude is then calculated by the difference of u_n and R_n, scaled by a factor A. The PSC decays with time constant τ_{PSC}.

2.2.2. Circuit implementation

In order to transform the iterative model to continuous-time, the exponential time dependence can be implemented with exponentially decaying voltage traces. These are generated by the circuit shown in **Figure 3** for the internal state variables u, R, and PSC, which model facilitation, depression and postsynaptic current trace, respectively. At incoming presynaptic spikes these decay traces are triggered and the resulting PSC amplitude is calculated by the difference of facilitation and depression value as shown in Equation 3. In **Figure 3** the circuit schematic is shown comprising three similar parts, for calculating V_U, V_R, and V_{PSC}.

When a presynaptic spike occurs these voltages are updated by a special switching scheme presented in **Figure 4**. V_U is increased toward V_A, which represents the global scaling factor A in Equation 3. The number of switching events of the V_U update determines the parameter U. α is set by the number of switching events of the V_R update. Switches S_{17} and S_{18} transfer the voltage difference of V_U and V_R to V_{PSC}.

Between incoming spikes an exponential decay of V_U, V_R, and V_{PSC} is performed by SC leaky integrator circuits. The working principle will be explained for the facilitation subcircuit and can be applied analogously for depression and PSC generation. On every decay event (see "Decay V_u" in **Figure 4**) C_{RU} (5 fF) is discharged in a first switching phase Φ_1 (see also bottom right of **Figure 3**). In this period C_U (75 fF), which stores the value of the facilitation variable, is fully decoupled from the circuit. Switching phase Φ_2 performs a charge equalization on C_U and C_{RU}. Thus, on every decay event V_U is decreased by a factor $\frac{C_U}{C_U + C_{RU}} = \frac{15}{16}$. These decay events are repeated with period T_u. With $\frac{15}{16} = \exp(-\frac{T_u}{\tau_u})$ we can easily calculate T_u for a desired decay time constant τ_u:

$$T_u = -\tau_u \cdot \ln\left(\frac{15}{16}\right) \approx \tau_u \cdot 0.0645. \tag{4}$$

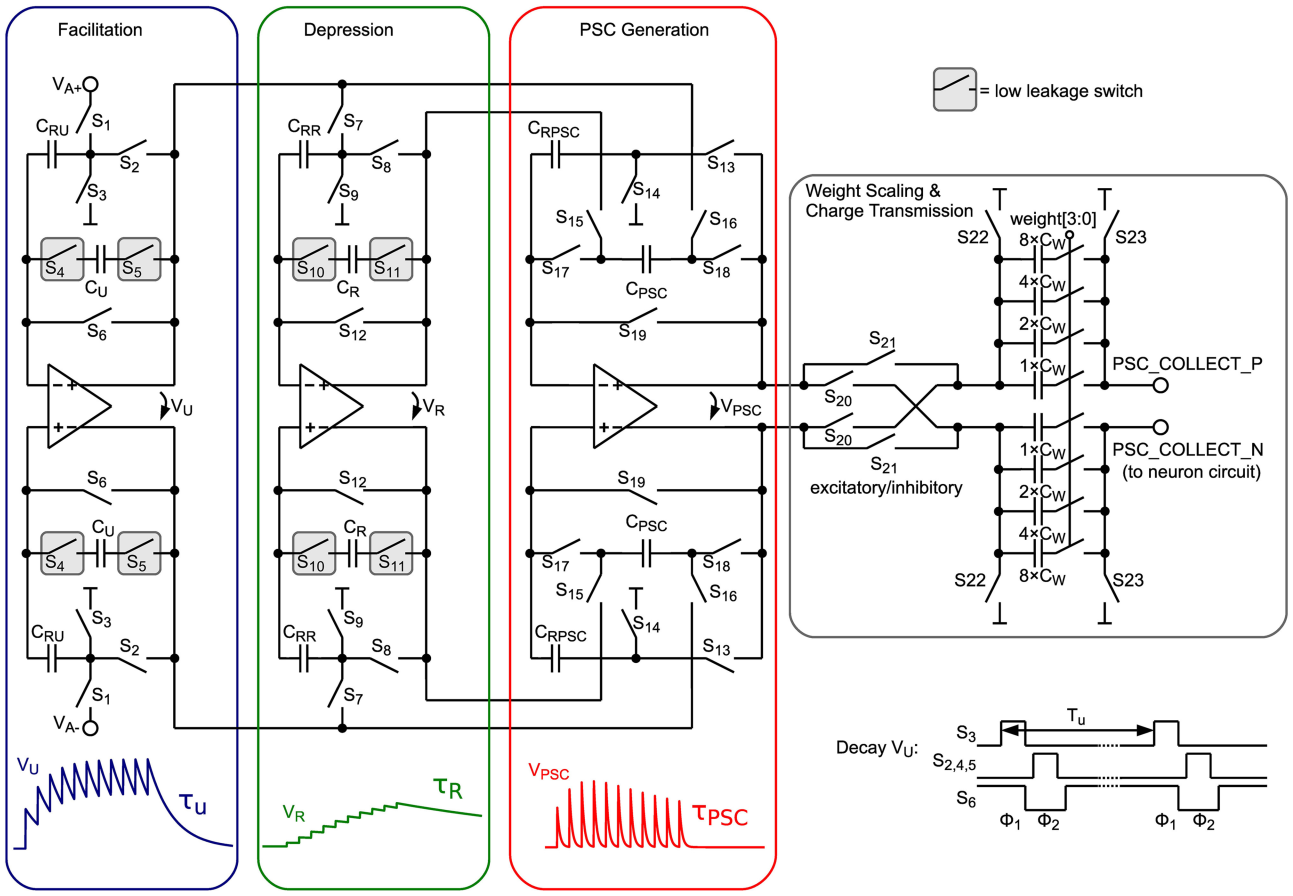


FIGURE 3 | Schematic of the presynaptic adaptation circuit comprising 3 fully-differential SC leaky integrator circuits. Capacitors storing the value of the corresponding model variables are encapsulated by dedicated low-leakage switches.

Since T_u is derived from a digital counter driven by the system clock, τ_u is proportional to the counter size and system clock frequency and allows to set time constants ranging from a few milliseconds to about one second. In order to scale the system's overall speed there is a tunable system clock divider, which enables to operate the circuit from biological real-time up to a 100-fold acceleration, keeping all relative timings without the need for adjusting bias voltages.

With the period of the matrix column cycle, the resulting exponentially decaying PSC voltage is sampled on the 4-bit binary-weighted capacitor C_W and transferred to the neuron circuit.

2.2.3. Leakage reduction

The maximum achievable time constant is limited by subthreshold leakage and junction leakage in the switches (see I_1 and I_2, resp. in **Figure 5B**) (Roy et al., 2003). A dedicated technique similar to Ellguth et al. (2006) and Ishida et al. (2006) has been applied for switches surrounding capacitors C_U and C_R where the switch transistor is split into two transistors (see **Figure 5A**). If the switch is in off-state the middle node V_M is clamped to a fixed voltage V_{LL}. Switch signals S and S_{LL} are non-overlapping. With $V_{LL} = 250\,\text{mV}$, which is equal to the common-mode voltage, drain-source voltage of M1 and M2 is kept low, which minimizes subthreshold leakage. Furthermore, the amount of leakage current is independent of the voltage at the other switch terminal. Junction leakage is minimized by minimal sized drain and source terminals. With a reduced voltage swing of about $V_{DD}/2$ all switches can be implemented with NMOS transistors only, which keeps leakage currents low and reduces circuit complexity. Especially the concept of isolating capacitors by low-leakage switches makes it possible to reach time constants up to 600 ms, which is the maximum controllable setting in our design, despite using small capacitance values in the 28 nm technology node (which naturally has high leakage). This is demonstrated by the measurements in Section 3.2. Thus, we achieve an off-resistance of about 600 ms/75 fF = 8TΩ, which corresponds to a conductance of 125 fS. In contrast to another technique recently proposed by Rovere et al. (2014), which requires two auxiliary low offset opamps, our solution is much more area and power efficient and satisfies our leakage constraints.

2.2.4. Proposed opamp

For buffering V_u, V_R, and V_{PSC} a two stage opamp is used (see **Figure 6**), since transistor stacking is difficult at supply voltages of 1 V. A gain boosting technique similar to Dessouky and Kaiser

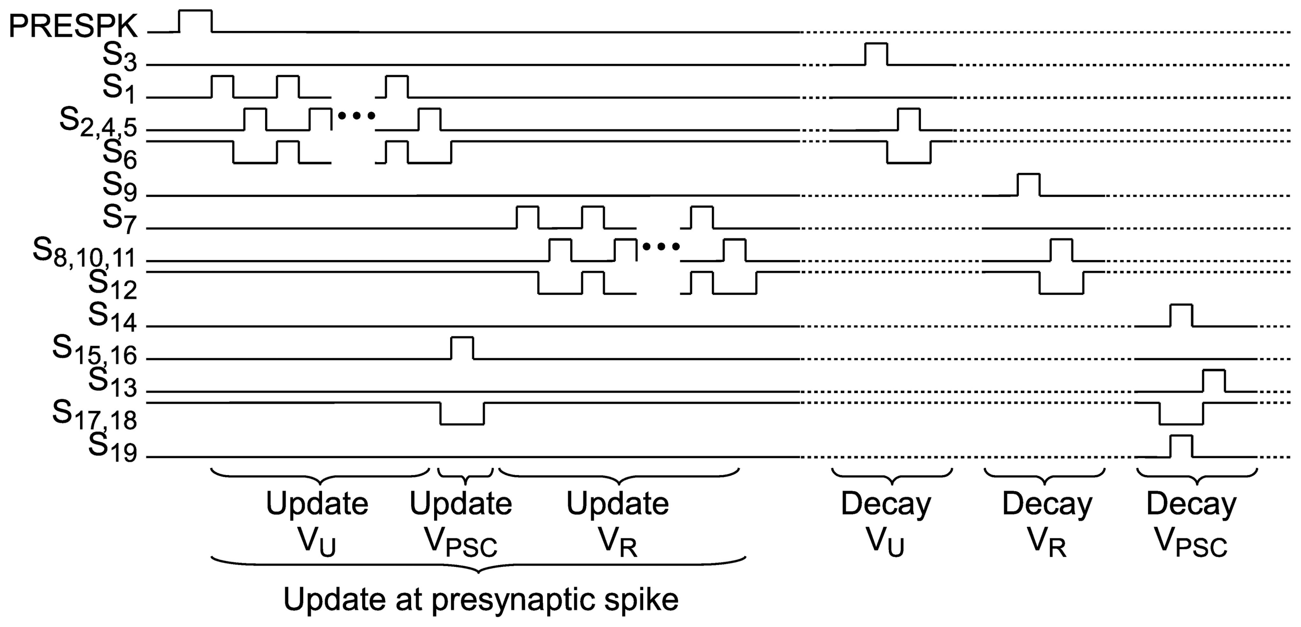


FIGURE 4 | Switch signals for update at an incoming presynaptic spike and for exponential decays of V_U, V_R and V_{PSC}. Dotted lines indicate that decay events can occur independently as well as simultaneously.

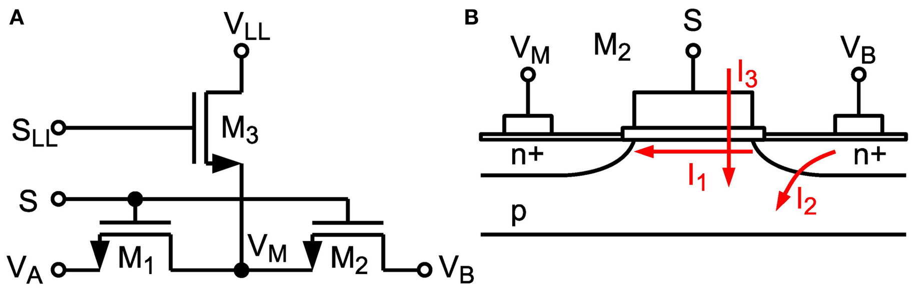


FIGURE 5 | (A) Low-leakage switch configuration. **(B)** Cross-section of MOS Transistor M2 with denoted subthreshold leakage (I_1), junction leakage (I_2) and gate leakage (I_3).

(2000) has been applied, where the load of the first stage has been split into two cross-coupled transistors (M_3, M_5 and M_4, M_6). By connecting the gates of M_5 and M_6 to the opposite output of the first stage a positive feedback is generated. The common-mode voltage of the first stage is well defined by the diode connected transistors M_3 and M_4 whereas the common-mode voltage of the output stage (M_7–M_{14}) is controlled by an SC CMFB network. In order to derive stability a classical miller compensation (C_1, R_1, C_2, R_2) has been applied using poly resistors and custom designed metal-oxide-metal capacitors. At the output an NMOS source follower ($M_{11} - M_{14}$) is connected, which enhances slew rate performance. Thus, the output voltage range is limited to 0–500 mV, which corresponds to the allowed voltage range of the low-leakage switches. The input common mode voltage range is 0–420 mV, which is sufficient for $V_{cm} = 250$ mV. The opamp consumes an area of 68 μm^2 and achieves an open-loop gain of 54 dB. It is designed to operate in biological real-time, as well as in a 100-fold accelerated environment. In fast mode the opamp draws 30 μW of power and has a slew rate of 60 V/μs. As the capacitor settling time scales with speed-up, the power consumption in real-time operation can be reduced by a factor of 100, i.e., down to 300 nW.

2.2.5. *Offset compensation*

Due to the small area occupied by the opamp, which is important for large scale integration, mismatch results in a maximum input offset voltage of about ±16 mV. Nevertheless, this offset can be compensated by a simple auto-zeroing technique (Enz and Temes, 1996). As can be seen in **Figure 3**, in the sampling phase (Φ_1) input voltages and common-mode voltages, respectively, are sampled against virtual ground of the opamp (switches S_6, S_{12} and S_{19} are closed). Since the offset voltage is present at the opamp input at this time, it is also sampled, and thus, canceled out at the output in the second phase (Φ_2). Despite the existence of more

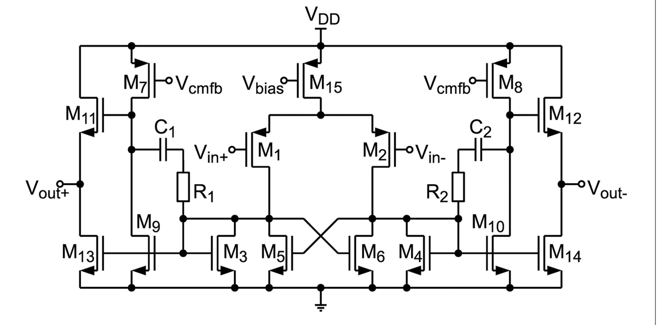


FIGURE 6 | Proposed opamp circuit used for buffering V_u, V_R, and V_{PSC}.

advanced auto-zeroing techniques in the literature, this technique has been chosen, because neither additional capacitors nor additional switching phases are required, reducing area and circuit complexity.

2.3. SWITCHED-CAPACITOR IMPLEMENTATION OF A BISTABLE STOCHASTIC SYNAPSE

2.3.1. Model

The stop learning model of long-term plasticity has been introduced in Brader et al. (2007), based on earlier work in Fusi et al. (2000). The model represents a synapse with two stable states, potentiated and depressed, whereby the state transition between both stable states is regulated via a continuous internal state X(t) of the synapse. X(t) is influenced by a combination of pre- and postsynaptic activity, namely the presynaptic spike time t_{pre} and the value of the neuron membrane voltage $V_{mem}(t)$. A presynaptic spike arriving at t_{pre} reads the instantaneous values $V_{mem}(t_{pre})$ and $C(t_{pre})$. The conditions for a change in X depend on these instantaneous values in the following way:

$$\mathrm{X} \rightarrow \mathrm{X} + a \quad \text{if} \quad \{V_{mem}(t_{pre}) > \theta_V \quad and \quad \theta_{up}^l < C(t_{pre}) < \theta_{up}^h\} \tag{5}$$

$$\mathrm{X} \rightarrow \mathrm{X} - b \quad \text{if} \quad \{V_{mem}(t_{pre}) \leq \theta_V \quad and \quad \theta_{down}^l < C(t_{pre}) < \theta_{down}^h\}, \tag{6}$$

where a and b are jump sizes and θ_V is a voltage threshold. In other words, X(t) is increased if $V_{mem}(t)$ is elevated (above θ_V) when the presynaptic spike arrives and decreased when $V_{mem}(t)$ is low at time t_{pre}. The θ_{up}^l, θ_{up}^h, θ_{down}^l, and θ_{down}^h are thresholds on the calcium variable. The calcium variable C(t) is an auxiliary variable (see Brader et al., 2007 for details) that provides a low-pass filter of the postsynaptic spikes. This gives the ability to stop the learning based on thresholded, long-term averages of postsynaptic activity. In the absence of a presynaptic spike or if stop learning is active [i.e., C(t) hits the respective threshold], then X(t) drifts toward one of two stable values:

$$\frac{d\mathrm{X}}{dt} = \alpha \qquad \text{if} \qquad \mathrm{X} > \theta_X \tag{7}$$

$$\frac{d\mathrm{X}}{dt} = -\beta \qquad \text{if} \qquad \mathrm{X} \leq \theta_X \tag{8}$$

The bistable state of the synapse is determined according to whether X(t) lies above or below the threshold θ_X. Computationally, this model is interesting because through X(t) it can learn a graded response to an input pattern even though the output weight of the synapses is binary. The model also has some biological veracity, being sensitive to pre-post and post-pre spike patterns in a manner similar to the well-known spike time dependent plasticity (Brader et al., 2007).

2.3.2. Circuit implementation

The circuit schematic shown in **Figure 7** replicates the model described in Equations (5–8). In contrast to the circuit presented in (Indiveri et al., 2006) our implementation makes use of SC technique. Thus, the model equations are solved in a time-discrete fashion, which enables the use of low-leakage switches as shown in Section 2.2.3 to achieve very low drift rates α and β. The time-discretization also allows for time multiplexing the single synapse circuits, thus, one driver circuit (see blue box in **Figure 7**) can drive multiple (in our case 64) synapses (red boxes). Due to the removal of active elements, one synapse circuit can be reduced to only 2 capacitors and 4 low-leakage switches storing the synapse state X (cp. Equations 5–8) as a differential voltage. The synapse occupies an area of 3.6 μm × 3.6 μm which is shared equally by the two synapse capacitors with 22 fF each. These are custom-designed metal-oxide-metal capacitors, utilizing an interdigitated fingered layout in the complete 5-layer metal stack with cut-outs on the lower two layers for wiring. The low-leakage switches are located directly below the capacitors. Each synapse can be connected to the driver circuit via switches $S_{syn,i}$, where i indicates the column number in the synapse matrix, and 4 wires V_{INP},V_{INN},V_{XP}, and V_{XN}. The driver circuit is basically an SC integrator, which integrates different voltages V_α, V_β, V_a, and V_b in dependence of synapse state, neuron state and incoming presynaptic spikes onto the synapse capacitors $C_{syn,i}$. The integrator's opamp is the same as for the presynaptic driver presented in Section 2.2.4. As shown in the timing diagram in the lower right corner of **Figure 7**, the operation principle can be divided into 4 phases "Reset," "Readout," "Comparison" and "Integration" for one synapse. All synapses of one row are cycled through sequentially, whereas all rows are processed in parallel.

In the reset phase an offset compensation of the opamp (cp. Section 2.2.5) is performed, which avoids the integration of a possible offset voltage as well as residual charge on the relatively long wires to the synapses. Therefore, switches annotated with Φ_{reset} are closed, which closes a negative unity-gain feedback loop around the opamp. The offset voltage appearing at the opamp input is then stored on capacitors C_{refr} and C_{hebb} and can be subtracted in the integration phase.

After reset a readout of the synapse state is performed. Switches $S_{syn,i}$ of the currently active synapse i are closed, which places the synapse capacitors in the feedback path of the opamp. The voltage stored on the capacitors, i.e., the synapse state X, is now visible at the opamp output between the differential lines V_{XP} and V_{XN}.

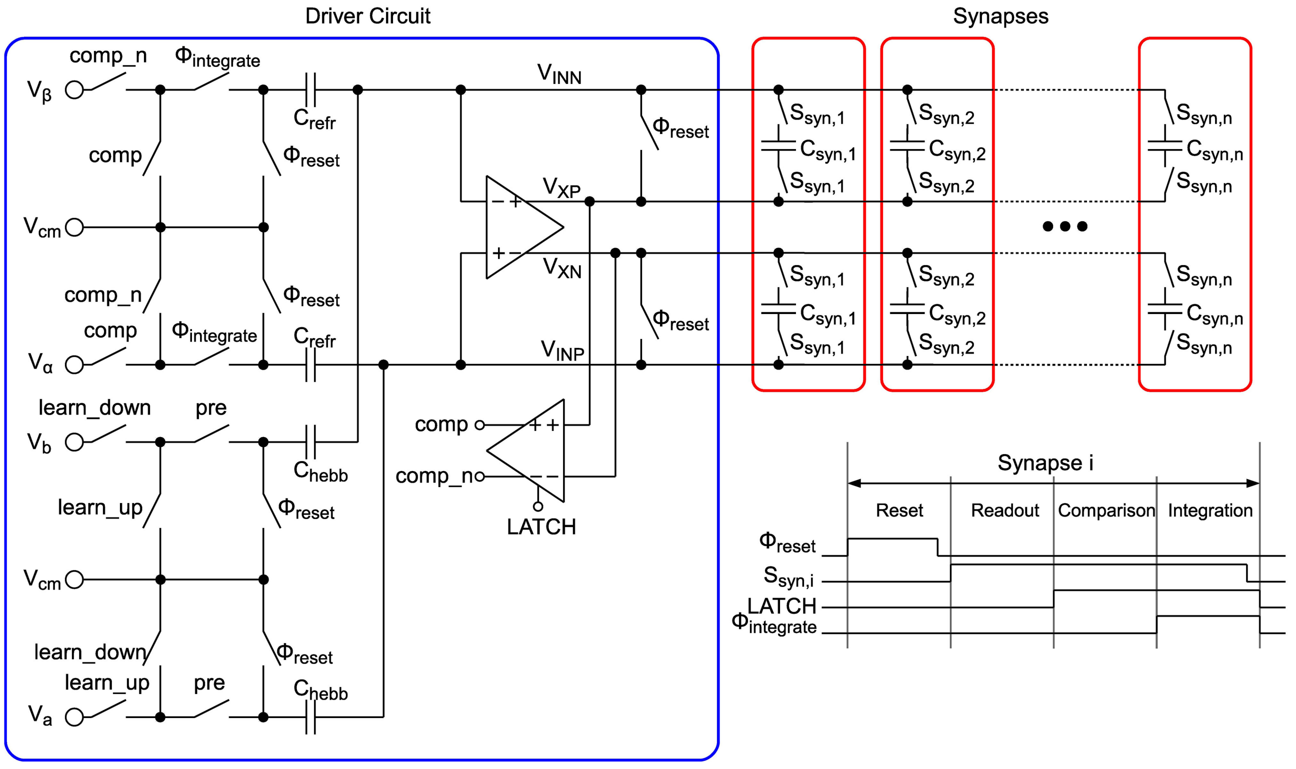


FIGURE 7 | LTP circuit.

When the readout is completed the synapse capacitors stay connected and a comparison of the synapse state with threshold Θ_X is performed. In the implementation Θ_X is fixed at 0.5, thus, the comparator (see Section 2.3.3) only has to compare whether $V_{XP} > V_{XN}$. After comparison the result is provided by signals comp and its inverted counterpart comp_n.

In the integration phase the refresh part (see Equations 7, 8) and the hebbian part (Equations 5, 6) of the learning model are performed. In this phase switches annotated with $\Phi_{integrate}$ are closed. If comp is high then the differential synapse voltage V_X is increased by $\frac{C_{refr}}{C_{syn}} \cdot (V_\alpha - V_{cm})$, otherwise it is decreased by $\frac{C_{refr}}{C_{syn}} \cdot (V_\beta - V_{cm})$. This results in refresh rates of

$$\alpha = \frac{C_{refr}}{C_{syn}} \cdot \frac{(V_\alpha - V_{cm})}{\Delta t} \tag{9}$$

and

$$\beta = \frac{C_{refr}}{C_{syn}} \cdot \frac{(V_\beta - V_{cm})}{\Delta t}, \tag{10}$$

where $\Delta t = 0.62$ ms, which is the time needed for processing the 64 synapses of a row sequentially (in biological real-time mode).

If a presynaptic input spike arrives, then switch signal pre is high during the integration phase. In dependence of the post-synaptic membrane state Θ_V signals learn_up and learn_down are set. The neuron circuit providing the membrane state is an SC leaky integrate-and-fire neuron presented in the companion paper Mayr et al. (in press). It is equipped with two comparator circuits for spiking threshold detection and for judging the current membrane state, i.e., the $V_{mem}(t_{pre}) \gtrless \theta_V$ condition of Equation (5) resp. Equation (6). If $V_{mem}(t_{pre}) > \theta_V$, then learn_up is high and learn_down is low (neglecting the "stop learning" mechanism for now). Thus, the upward jump size is calculated by

$$a = \frac{C_{hebb}}{C_{syn}} \cdot (V_a - V_{cm}). \tag{11}$$

If $V_{mem}(t_{pre}) < \theta_V$, then learn_up is low and learn_down is high, which results in the downward jump size of

$$b = \frac{C_{hebb}}{C_{syn}} \cdot (V_b - V_{cm}). \tag{12}$$

In order to reduce the number of control voltages, single-ended input voltages are provided. The resulting common mode offset, caused by this asymmetry, is compensated by the SC CMFB circuit.

The "stop learning" feature described in Section 2.3.1 is handled by setting learn_up resp. learn_down to low using combinational logic (not shown). Therefore, the state of the calcium variable can be calculated externally in an FPGA, where the post-synaptic spike train is filtered by a low pass filter. The low pass filter output is then compared against the stop learning thresholds θ^l_{up}, θ^h_{up}, θ^l_{down}, and θ^h_{down} and the two resulting binary signals for enabling learning in the up and down direction, respectively, are transmitted to the driver circuit. As an additional feature for testing we implemented a "learn force" mode where learn_up and learn_down can be set explicitly, similar to keeping the neuron membrane permanently elevated or depressed.

The comp signal, which is provided in the "Comparison" phase states whether the synapse is depressed (LTD) or potentiated (LTP). This binary output is used to scale the PSC generated by the presynaptic adaptation circuit (see "Weight Scaling and Charge Transmission" in **Figure 3**). Therefore, each synapse has two 4-bit weights for LTP and LTD stored in a RAM (see **Figure 1**), which is chosen accordingly to the synapse state and transmitted to the weight scaling circuit. The scaling of the PSC is done via binary weighted capacitors, transferring charge to the neuron circuit. Additionally each synapse is selectable excitatory or inhibitory, which inverts the PSC voltage. Thus, inhibitory stop-learning synapses are also possible.

2.3.3. Comparator circuit

A circuit schematic of the comparator shown in **Figure 7** is depicted in **Figure 8A**. It consists of a preamplifier (see **Figure 8B**), which is inspired by Dessouky and Kaiser (2000) and a simple dynamic latch circuit (Song et al., 1995) shown in **Figure 8C**. This architecture has been chosen, because the dynamic latch circuit can have a high random offset voltage of up to 20 mV, caused by mismatch. The preamplifier raises the differential signal level to minimize decision errors, caused by this mismatch. The preamplifier is therefore equipped with an offset compensation (compare Section 2.2.5). At the output of the comparator circuit an SR-latch is connected, which stores the result until the next comparison.

2.4. MEASUREMENT SETUP AND CHARACTERIZATION METHODS

As detailed in Section 2.1, the entire system is ratiometric with respect to the clock frequency. That is, the system clock can be scaled so that the neuromorphic system operates anywhere from biological real time up to a factor 100 faster. As operation at biological real time is the most challenging in circuit terms as well as the most interesting in terms of computation, real-time operation was used for the measurements in this paper. The corresponding clock frequency is 3.3 MHz, generated by a configurable clock divider from the 330 MHz central system clock. At this frequency, the synaptic matrix update period is 0.62 ms (compare Section 2.1).

As the different leakage currents of MOS switches are highly temperature dependent, we investigated how well our low-leakage switch technique operates at different temperatures. Thus, the measurements of the presynaptic adaptation are carried out at the temperatures indicated by using the temperature controlled setup shown in **Figure 9**. The IC package is held at the adjusted temperature with ca. ±2 °C deviation. The output of the presynaptic adaptation can be measured either via tracing the PSC time course from one of the analog test outputs or indirectly by monitoring the spike output of a connected neuron. Directly measuring the PSC voltage via an oscilloscope is well-suited for detailed short-time measurements, which we used to verify correct operation of the circuitry. For reducing noise in this case, the aquired waveform data was averaged over time bins of 0.1–0.3 ms.

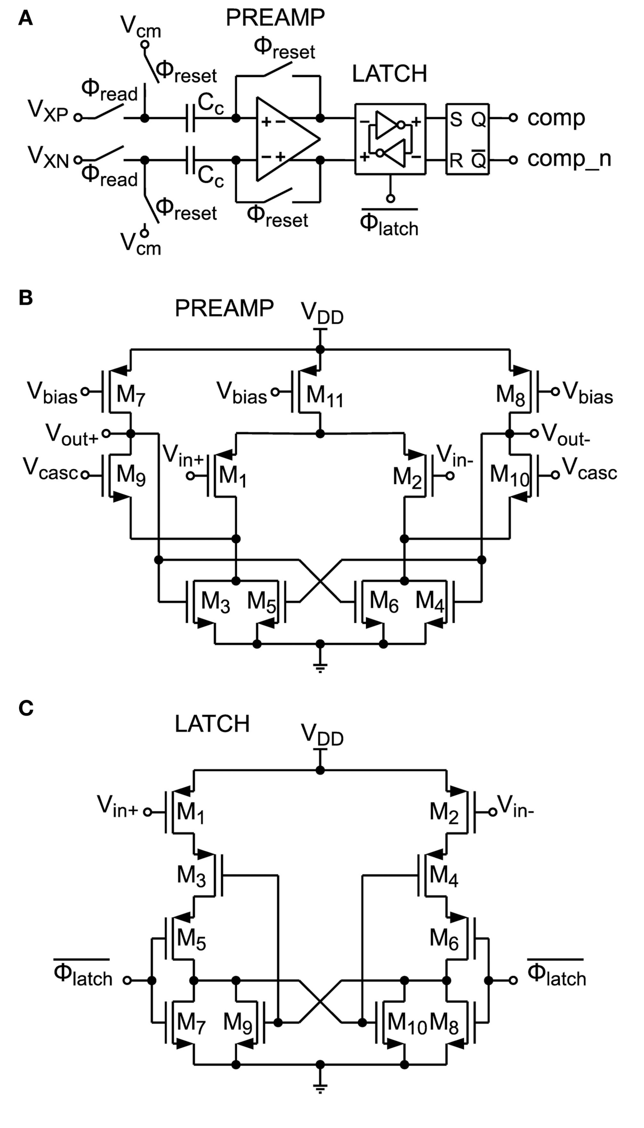


FIGURE 8 | (A) Comparator circuit with offset-compensated preamplifier, compensation capacitors C_c and latch circuitry. **(B)** Preamplifier circuit schematic. **(C)** Latch circuit schematic.

Direct oscilloscope measurements are less practical for automatic extraction of a multitude of time constants. For this case, we used the following purely spike-based protocol: The adaptation state is probed by sending an input spike and counting the number of output spikes in reaction. For getting a reasonably strong response, the synaptic weight and the PSC scaling voltage are set to their maximum values. Setting the membrane time constant to a high value as well, the number of output spikes per input spike is approximately linearly dependent on the PSC amplitude. For the measurements, we only activated depression, so that the PSC amplitude of a spike directly resembles the current state of the depression variable. For each time constant measurement, the depression variable is charged by initially applying 10 spikes. Afterwards, the adaptation strength is set to zero, so that the depression variable relaxes back to its resting state. This relaxation

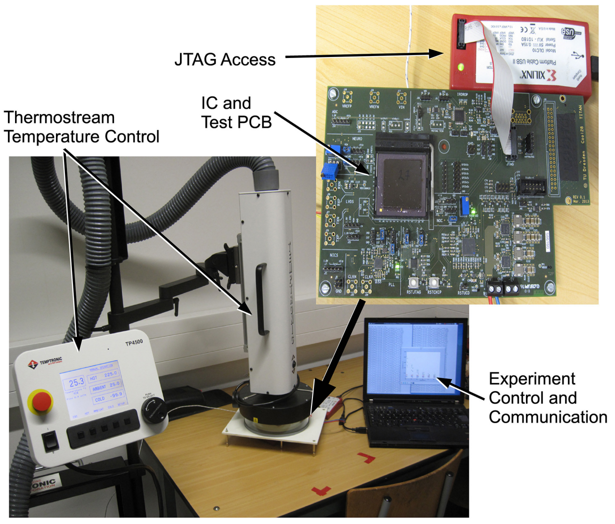


FIGURE 9 | Setup for measurements with controlled temperature.

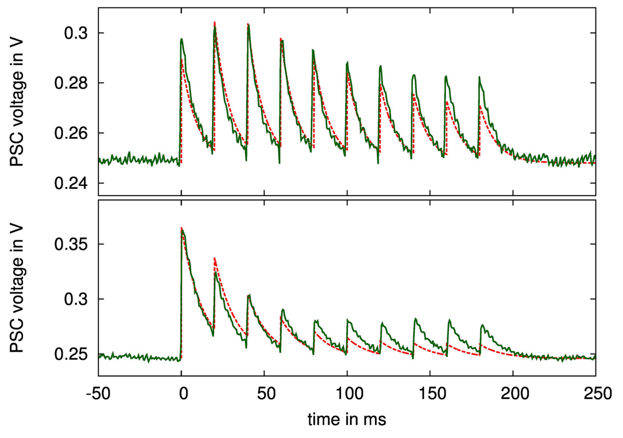


FIGURE 10 | PSC voltage traces of a simultaneously facilitating and depressing (top), and of a depressing (bottom) synapse when stimulated with 10 spikes at 50 Hz rate. Configuration parameters: top: $\tau_u = 300$ ms, $\tau_R = 300$ ms, $\tau_{PSC} = 10$ ms, $U = 0.29$, $\alpha = 0.5$, bottom: $\tau_u = 10$ ms, $\tau_R = 490$ ms, $\tau_{PSC} = 13$ ms, $U = 0.96$, $\alpha = 0.5$. The nominal time courses for the PSC voltages with these parameters and fitted amplitudes are drawn as dashed lines.

is monitored by continuously probing the state with input spikes. From the relaxation time course, the time constant is extracted by calculating the best-fitting (smallest root mean squared error) exponential function, with amplitude and time constant as free parameters. Results are averaged over 10 repetitions.

The measurements of the stop learning synapses are carried out at ambient temperature, i.e., no special measures for chip cooling are taken.

3. RESULTS

3.1. BASIC OPERATION OF THE PRESYNAPTIC ADAPTATION

For evaluating the presynaptic adaptation performance, we stimulated a presynaptic circuit with a regular spike train for two different adaptation types, as shown in **Figure 10**. We chose a parameter set for combined facilitation and depression to demonstrate correct operation of the circuit as a whole, and a setting for a depressing synapse, where the depression variable dominates the behavior. The latter case is used for assessing the correct reproduction of long time constants in the next section.

Figure 10 also shows ideal time courses for the implemented model with the same parameters and fitted amplitude and offset. The measurements agree well with these nominal curves even without calibrating any parameters. They differ mainly in the adaptation strength, i.e., in the ratio between highest and lowest PSC amplitude, which is smaller in the measured curves. This may be caused by time constants being too small, or by charge injection effects, resulting in voltage offsets during updates of the adaptation variables at incoming spikes.

3.2. CHARACTERIZATION OF THE PRESYNAPTIC ADAPTATION TIME CONSTANTS

Figure 11 shows traces over different time constant settings for one presynaptic adaptation circuit. The time course of the depression relaxation for nominal settings as well as with only leakage present can be faithfully fitted by an exponential function, allowing for calculation of the depression time constant.

Measured time constants of 16 adaptation circuits from 4 chips are shown in **Figure 12**. The values are well-controlled in the configurable range up to 300 ms at all temperatures with sigma less than 15% and the mean within 20% of the nominal setting. The same is true for the 600 ms setting up to 30°C. Above that, the leakage influence causes the measured mean to be at least one sigma outside the nominal, which constitutes our fail criterion.

Using the infinite setting for the depression time constant, i.e., there are no decay switching events, this leakage can be measured, see upper plot in **Figure 12**. As expected, it is highly temperature-dependent. For temperatures of 30°C and below, all measurements are above 1 s, so that time constants up to this value are feasible at room temperature if the controlled leakage, i.e., the switching frequency of the decay process, is further decreased compared to the 600 ms setting. As described in Section 2.2.3, a time constant of 600 ms corresponds to a leakage resistance of 8 TΩ. This value increases to a minimum of 13 TΩ for time constants of 1 s or above. These high resistances demonstrate the effectiveness of the employed leakage reduction techniques.

The measurements show that time constants of several seconds are possible at temperatures below 30°C. As the time constants caused by intrinsic leakage show a larger spread for these temperatures, individual calibration of the switching frequency for the leakage mechanism may be required to still achieve well-controlled time constant values. Nevertheless, for the envisaged time constant range up to 600 ms of the design, the measurements demonstrate correct resemblence of time constant values at room temperature, so that all further measurements were performed without any special measures for temperature control.

3.3. CHARACTERIZATION OF THE BISTABLE STOCHASTIC SYNAPSE

In this section, results for the SC implementation of the stop-learning synapse are given. As detailed in Section 2.3.2, a force bit can be set that forces the synapse to transition from potentiated

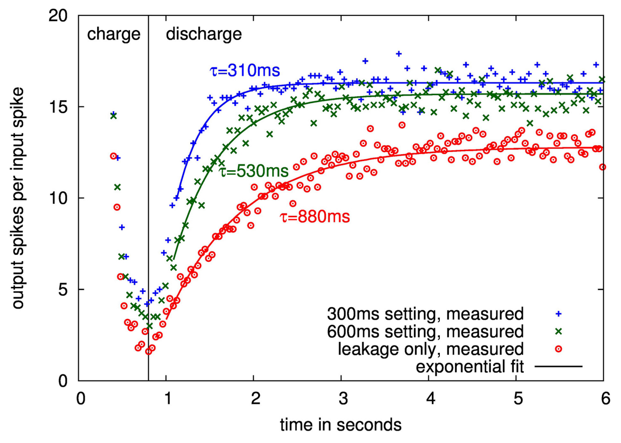


FIGURE 11 | Measured time courses of input-output gain for one presynaptic adaptation circuit at 40°C with 300 ms, 600 ms and leakage only settings. Time course until 0.8 s is the charging of the depression, following, the synapse relaxes back to its steady state with the depression time constant.

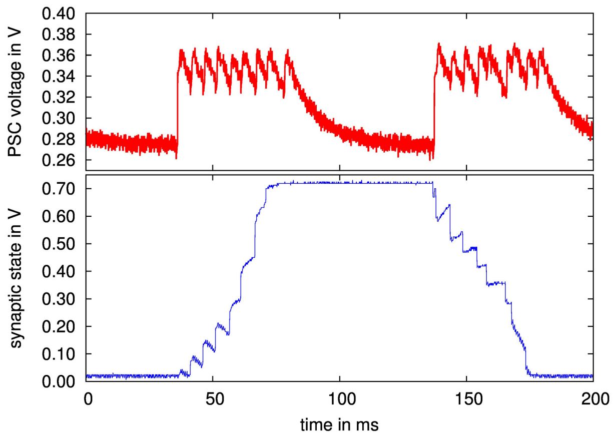


FIGURE 13 | (Upper diagram) Measured PSC waveform of a 200 Hz presynaptic spike train with 12 pulses; (lower diagram) synapse state of stochastic stop learning synapse, with forced transition from depressed to potentiated state and back.

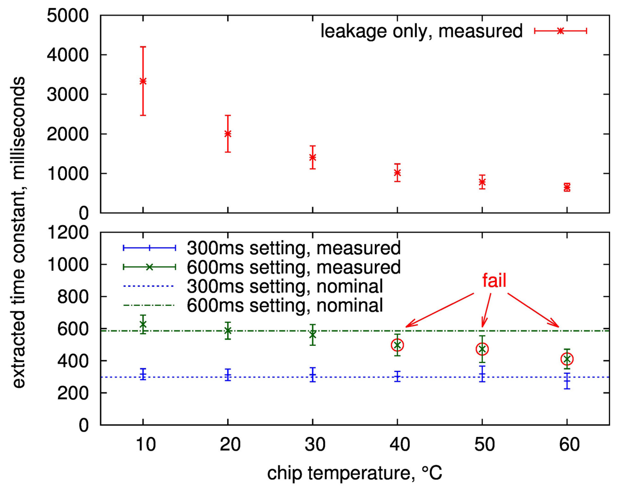


FIGURE 12 | Mean and standard deviation (error bars) of extracted time constants over 16 presynaptic adaptation circuits of four separate ICs. Shown is the measured time constant for a setting of infinity (upper part, i.e., the equivalent time constant if just leakage is active) and two configured time constants (nominal 600 and 300ms) for the presynaptic adaptation circuit of **Figure 3**.

to depressed state or vice versa. That is, Equation 5 resp. Equation 6 are forced to always employ a or b, similar to setting $V_{mem}(t)$ either to a constant high or low value. A presynaptic spike train of 12 spikes is then applied to the synapse, as shown in the upper diagram of **Figure 13**.

From the lower diagram of **Figure 13**, it can be observed that the synapse reaches a stable potentiated state (at ca. 0.7 V) or a depressed state (at 0 V). For the transition at 50 ms, the force bit activates only a, forcing the synapse to become potentiated. Conversely, at 150 ms, only b is active, the synapse becomes depressed. Between presynaptic events, the curve shows that α and β draw the synapse back to one of its stable states, according to the synapse state being above or below θ_X (set at half way between the two stable states, see also Equation 7 resp. 8).

To test the stop learning functionality expressed in our implementation by the two stop learning bit flags (see Section 2.3.2), a second experiment is carried out. The packet of 12 presynaptic spikes is split in two parts which are sent immediately after each other, see the corresponding PSC voltage in the upper diagram of **Figure 14**. Starting from the depressed state, the force bit activates a, but after the first part of the presynaptic spike packet, which contains 6 pulses, the stop learning bit for a is activated. This causes the last 6 pulses to be discarded in terms of synaptic state modification, i.e., only β is active which draws the synapse back down to the depressed state.

At 150 ms, this experiment is repeated, but the stop learning is activated after 8 pulses. This is sufficient to push the synapse above θ_X, i.e., α becomes active which draws the synapse state to the potentiated state, even though the last 4 presynaptic pulses are again discarded because of the activated stop learning. Thus, overall functionality of the stochastic stop learning synapse is confirmed. In this experiment, the stop learning was set explicitely. As stated in Section 2.3.2, the future backplane for a multi-chip system will compute the Calcium variable externally on an FPGA based on the output spike rates (Brader et al., 2007), setting the stop learning bits dynamically based on the Calcium state.

Please note that we are only showing the internal synaptic state transitions. For the overall network dynamics, the state change means a switch between the 4 bit potentiated and 4 bit depressed weights (compare Section 2.3.2). Thus, while learning induction is in the form of the one bit decision of the original stop learning synapse (Brader et al., 2007), the expression of the synaptic learning can be individual for each synapse, adding significantly to network richness compared to the global settings for potentiated and depressed synapses in other implementations of this plasticity rule (Indiveri et al., 2006). This capability for individual weights could also be exploited for implementations of the Neural

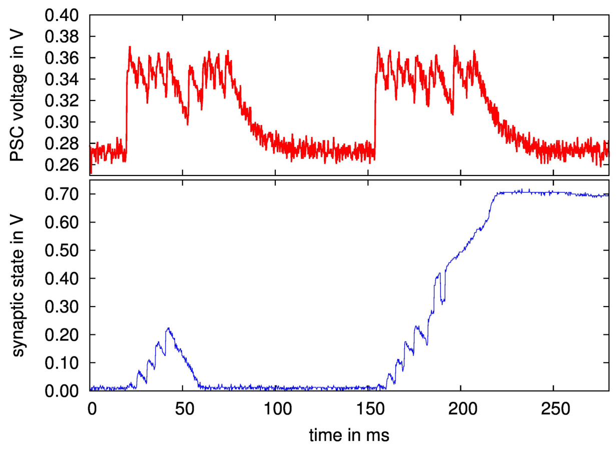


FIGURE 14 | (Upper diagram) Measured PSC waveform of presynaptic spike train, both packets 12 pulses, 200 Hz; (lower diagram) synapse state of stochastic stop learning synapse, with forced transition from depressed to potentiated state. The first transition is aborted due to activation of stop learning after 6 pulses, i.e., at a point where the synapse state is not above θ_X and thus gets drawn back to the depressed state. For the second transition, stop learning is activated after 8 pulses.

Table 1 | Characteristics of the presented SC neuromorphic system.

Technology	**Global foundries 28 nm SLP**
Layout area for system	460*430 μm^2 neuromorphic comp., 600*600 μm^2 overall (including DAC, RAM, etc.)
Clock frequency	330 MHz (PLL), 3.3 MHz (neuromorphic components)
VDD analog	1.0 V
VDD digital	0.75 V
Power digital	1.1 mW (speed-up 1) to 3.1 mW (speed-up 100)
Power analog (neuromorphic components)	0.38 mW (speed-up 1) to 11.0 mW (speed-up 100)
Power analog (PLL)	0.45 mW
Neuron model	LIAF (Rolls and Deco, 2010)
Presynaptic adaptation	Facilitation and depression (Noack et al., 2012)
Synaptic plasticity	Stochastic synapse with stop learning (Brader et al., 2007)
System characteristics	128 presynaptic adaptation circuits, 8192 stochastic synapses, 64 LIAF neurons

All figures are for a speed-up of one, i.e., biological real time operation, if not stated otherwise.

Engineering Framework (Eliasmith and Anderson, 2004) on our neuromorphic system. A 4 bit weight resolution plus the capability for setting each synapse excitatory or inhibitory should be sufficient for sophisticated population-based signal processing (Mayr et al., 2014), compare also the results achieved for 58 neurons with 4 bit synaptic weights in Corradi et al. (2014).

3.4. OVERALL RESULTS

Table 1 details the major characteristics of the neuromorphic system (Mayr et al., in press). Its power budget is competitive with recent power-optimized digital or analog neuromorphic systems of similar size (Indiveri et al., 2006; Seo et al., 2011). The digital part includes 0.45 mW static power draw which is mainly due to the other components on this test chip, so putting the neuromorphic system on a chip by itself would improve power consumption by about 23% at biological real time operation. The current clocking setup features a constant-frequency PLL (Höppner et al., 2013) and a clock divider, which draw constant power irrespective of the speed up factor. To save power, this could be replaced with a variable-frequency PLL with frequency-dependent power draw (Eisenreich et al., 2009).

Plasticity models with time constants up to seconds have been shown for this SC implementation in 28 nm. Thus, reliable, controlled behavior fully in keeping with biological real time operation is possible. The efficacy of our chosen method for low-leakage capacitive state holding has been proven, with detailed analysis of the effect of temperature on achievable time constants. The characterization of the presynaptic time constants employs the entire signal pathway of the system (compare **Figure 1**), showing complete overall functionality.

Table 2 gives a comparison with other current implementations of presynaptic adaptation and/or synaptic plasticity. The synapse area of our implementation is among the lowest, with only the static 1 bit synapse of a digital synaptic array smaller in size. Especially, compared to fully analog implementations of stop learning (Indiveri et al., 2006), the SC approach and agressive scaling for the various capacitances allow an implementation of stop-learning that benefits from the technology shrink. As can be seen from the faithfulness of model replication in SC, this scaling can be achieved without compromising functional richness and accuracy. When accounting for technology node, the area consumption of the presynaptic adaptation is larger than e.g., Bartolozzi and Indiveri (2007) or Schemmel et al. (2010). This is due to the fact that our presynaptic adaptation aims at a very faithful reproduction of the model of Markram et al. (1998), necessitating complex, multi-stage computational circuits (see **Figure 3**). Specifically, our implementation is the only one offering concurrently operating facilitation and depression.

The shown architecture always connects an input via synapses to all neurons, corresponding to an all-to-all connectivity. This is the same architecture as used for example in memristive crossbar arrays Alibart et al. (2012); Mayr et al. (2012). The main advantage of this architecture in our design is that it allows to implement all parts of the synapse circuit that depend on the input only once per synapse row. This significantly reduces circuit area, reducing the synapse circuit to an analog storage element in our design. The efficiency gain comes at the price of reduced flexibility concerning connection topologies. All-to-all and comparable connection structures are well-suited, whereas sparse connectivity results in a high number of unused synapses in the matrix, making the architecture less efficient in this case, even when optimizing the mapping of networks to the hardware architecture Mayr et al. (2007); Galluppi et al. (2012). To improve the efficiency, i.e., the fraction of utilized synapses, also for low connection densities, more presynaptic input circuits than

Table 2 | Comparison of the presented short- and long-term plasticity circuits with other implementations from literature.

Ref.	Techn.	System area	Synapse area	Number of synapses	Synapse functionality	Pre-synapse area	Number of presynapses	Presynapse functionality
Seo et al., 2011; Merolla et al., 2011	45 nm	4.2 mm^2	1.6 μm^2	262 k	1-bit static synapses, Set externally	–	–	Not implemented
Park et al., 2014	90 nm	16 mm^2	15 μm^2	262 k	Log-domain conductance-based synapse, no plasticity	–	–	Not implemented
Mitra et al., 2006; Bartolozzi and Indiveri, 2007	350 nm	12 mm^2	1200 μm^2	8192	Stop learning	1360 μm^2	NA	Short-term depression
Schemmel et al., 2010; Schemmel, personal communication	180 nm	50 mm^2	150 μm^2	115 k	STDP	84 μm^2	14 k	Either short-term depression or facilitation
This work	28 nm	0.36 mm^2	13 μm^2	8192	Stop learning	432 μm^2	128	Concurrent short-term depression and facilitation

synapse rows can be implemented, while synapses are made to choose between several inputs (Noack et al., 2010; Schemmel et al., 2010). This would only slightly increase the complexity of the individual synapse circuits, while greatly increasing the flexibility of the architecture (Noack et al., 2010).

4. DISCUSSION

4.1. PLASTICITY MODELS

Results show faithful implementation of the chosen short-term plasticity model (Markram et al., 1998). The detailed reproduction of this model endows the neuromorphic system with a corresponding rich behavioral repertoire, which could be employed for e.g., reproduction of population dynamics in cultured neurons (Masquelier and Deco, 2013) or simulation of short-term memory (Rolls et al., 2013).

The long-term plasticity rule is also reproduced well, opening up a host of information-theoretic applications, such as studies of memory retention, information content or classification performance of a network (Brader et al., 2007). Other flavors of long-term plasticity rules could also be supported by our neuromorphic system. For instance, the faithful reproduction of neuronal waveforms evident in **Figure 10** and their excellent configurability in terms of the time window (**Figure 12**) could also be employed for a plasticity rule based on neuron and synapse waveforms such as (Mayr et al., 2010), which aims at the replication of a wide range of biological plasticity experiments (Mayr and Partzsch, 2010).

4.2. SWITCHED-CAPACITOR NEUROMORPHICS

Dating back to Carver Mead, subthreshold CMOS has been the mainstay of neuromorphic circuit design, as it offers the advantage of low power consumption, ion-channel like behavior in CMOS devices and currents small enough to reach biological real time operation. However, such a fully analog implementation suffers from mismatch and leakage currents which are increasingly prevalent in deep submicron processes. In addition, the channel-to-transistor design philosophy means that this type of neuromorphic circuit consists largely of handcrafted circuits that depend crucially on the performance of each single transistor. Thus, porting a design between technology nodes essentially means a completely new design.

Switched-capacitor neuromorphic circuits move from this device level philosophy to a building block approach, i.e., the required model behavior is achieved with a combination of standard building blocks. SC is used as a mathematical framework to directly translate state-driven models to a mixed-signal realization. This keeps the neuronal states analog for biological veracity, while achieving significantly easier technology porting, as the circuit consists solely of standard building blocks such as amplifiers, switches and charge addition/subtraction. Representation of analog states at block level also eases implementation in deep submicron, as this takes advantage of the available device count for improved signal fidelity, while relying less on the characteristics of individual transistors. This building block approach allows agressive scaling of the active analog components, while the digital part of the SC circuits naturally scales with the technology node. Overall scaling is ultimately limited compared to a purely digital system by the largely invariant capacitor sizes, but is still significantly better than conventional, more device- and analog-centric neuromorphic approaches. As shown, this approach has enabled our SC system to deliver the same computational density as a purely digital neuromorphic system in a deep-submicron technology (Seo et al., 2011), while its power budget is on par with subthreshold circuits (Indiveri et al., 2006). When combined with deep submicron pixel cells (Henker et al., 2007), a sophisticated visual processing pyramid could be implemented (König et al., 2002; Serrano-Gotarredona et al., 2009).

While SC makes neuromorphic circuits possible in principle in deep submicron, one major challenge is still the leakage currents. The leakage completely precludes subthreshold circuits, but it also affects the stored states of capacitors in SC technique, especially for the timescales necessary for biological real time operation. As shown, we have solved this general challenge for SC neuromorphic circuits with our low leakage switch architecture, reaching controllable time constants >100 ms at ambient temperature.

4.3. NANOSCALE CMOS AND NOVEL DEVICES

Novel nanoscale devices, such as memristors, offer the possibility of very high density neuromorphic synaptic matrices (Alibart et al., 2012; Shuai et al., 2013). However, they need corresponding high-density neuronal driver circuits in CMOS. Moving neuromorphic circuits to deep-submicron technologies as outlined in this paper would provide this capability, i.e., very low footprint neuron driver and receiver circuits that generate analog waveforms for memristor synaptic matrices (Mayr et al., 2012).

ACKNOWLEDGMENT

This work is partly supported by "Cool Silicon," the "Center for Advancing Electronics Dresden" and the European Union 7th framework program, project "CORONET" (grant no. 269459).

REFERENCES

Alibart, F., Pleutin, S., Bichler, O., Gamrat, C., Serrano-Gotarredona, T., Linares-Barranco, B., et al. (2012). A Memristive nanoparticle/organic hybrid synapstor for neuroinspired computing. *Adva. Funct. Mater.* 22, 609–616. doi: 10.1002/adfm.201101935

Bartolozzi, C., and Indiveri, G. (2007). Synaptic dynamics in analog VLSI. *Neural Comput.* 19, 2581–2603. doi: 10.1162/neco.2007.19.10.2581

Brader, J., Senn, W., and Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Cassidy, A., Andreou, A. G., and Georgiou, J. (2011). "A combinational digital logic approach to STDP," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on* (Rio de Janeiro: IEEE), 673–676.

Corradi, F., Eliasmith, C., and Indiveri, G. (2014). "Mapping arbitrary mathematical functions and dynamical systems to neuromorphic VLSI circuits for spike-based neural computation," in *IEEE International Symposium on Circuits and Systems (ISCAS)* (Melbourne), 269–272.

Dessouky, M., and Kaiser, A. (2000). "Very low-voltage fully differential amplifier for switched-capacitor applications," in *Circuits and Systems, 2000. Proceedings ISCAS 2000 Geneva. The 2000 IEEE International Symposium on,* Vol. 5 (Geneva), 441–444.

Eisenreich, H., Mayr, C., Henker, S., Wickert, M., and Schüffny, R. (2009). A novel ADPLL design using successive approximation frequency control. *Elsevier Microelectr. J.* 40, 1613–1622. doi: 10.1016/j.mejo.2008.12.005

Eliasmith, C., and Anderson, C. C. H. (2004). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA: MIT Press.

Ellguth, G., Mayr, C., Henker, S., Schüffny, R., and Ramacher, U. (2006). "Design techniques for deep submicron CMOS / case study delta-sigma-modulator," in *Dresdner Arbeitstagung Schaltungs-und Systementwurf*, Vol. 1 (Dresden), 35–40.

Enz, C., and Temes, G. (1996). Circuit techniques for reducing the effects of op-amp imperfections: autozeroing, correlated double sampling, and chopper stabilization. *Proc. IEEE* 84, 1584–1614. doi: 10.1109/5.542410

Folowosele, F., Etienne-Cummings, R., and Hamilton, T. (2009). "A CMOS switched capacitor implementation of the Mihalas-Niebur neuron," in *BioCAS* (Beijing), 105–108.

Fusi, S., Annunziato, M., Badoni, D., Salamon, A., and Amit, D. (2000). Spike-driven synaptic plasticity: theory, simulation, VLSI implementation. *Neural Comput.* 12, 2227–2258. doi: 10.1162/089976600300014917

Galluppi, F., Davies, S., Rast, A., Sharp, T., Plana, L. A., and Furber, S. (2012). "A hierachical configuration system for a massively parallel neural hardware platform," in *Proceedings of the 9th Conference on Computing Frontiers* (ACM) (Cagliari), 183–192.

Grande, L., and Spain, W. (2005). Synaptic depression as a timing device. *Physiology* 20, 201–210. doi: 10.1152/physiol.00006.2005

Hartmann, S., Schiefer, S., Scholze, S., Partzsch, J., Mayr, C., Henker, S., et al. (2010). "Highly integrated packet-based AER communication infrastructure with 3Gevent/s throughput," in *Proceedings of IEEE International Conference on Electronics, Circuits, and Systems ICECS10* (Athens), 952–955.

Henker, S., Mayr, C., Schlüssler, J.-U., Schüffny, R., Ramacher, U., and Heittmann, A. (2007). "Active pixel sensor arrays in 90/65nm CMOS-technologies with vertically stacked photodiodes," in *Proceedings IEEE International Image Sensor Workshop IIS07* (Ogunquit, ME), 16–19.

Hong, Z., and Melchior, H. (1984). Four-quadrant CMOS analogue multiplier. *Electron. Lett.* 20, 1015–1016. doi: 10.1049/el:19840691

Höppner, S., Haenzsche, S., Ellguth, G., Walter, D., Eisenreich, H., and Schüffny, R. (2013). A fast-locking ADPLL with instantaneous restart capability in 28-nm CMOS technology. *Circ. Syst. II Exp. Briefs IEEE Trans.* 60, 741–745. doi: 10.1109/TCSII.2013.2278123

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

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

Ishida, K., Kanda, K., Tamtrakarn, A., Kawaguchi, H., and Sakurai, T. (2006). Managing subthreshold leakage in charge-based analog circuits with low-VTH transistors by analog T- switch (AT-switch) and super cut-off CMOS (SCCMOS). *Solid State Circ. IEEE J.* 41, 859–867. doi: 10.1109/JSSC.2006.870761

Khachab, N., and Ismail, M. (1991). A nonlinear CMOS analog cell for VLSI signal and information processing. *Solid State Circ. IEEE J.* 26, 1689–1699. doi: 10.1109/4.98991

Kinget, P. R. (2005). Device mismatch and tradeoffs in the design of analog circuits. *Solid State Circ. IEEE J.* 40, 1212–1224. doi: 10.1109/JSSC.2005.848021

Koickal, T., Hamilton, A., Tan, S., Covington, J., Gardner, J., and Pearce, T. (2007). Analog VLSI circuit implementation of an adaptive neuromorphic olfaction chip. *IEEE Trans. Circ. Syst. I Regular Pap.* 54, 60–73. doi: 10.1109/TCSI.2006.888677

König, A., Mayr, C., Bormann, T., and Klug, C. (2002). "Dedicated implementation of embedded vision systems employing low-power massively parallel feature computation," in *Proceedings of the 3rd VIVA-Workshop on Low-Power Information Processing* (Chemnitz), 1–8.

Markram, H., Wang, Y., and Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. *Proc. Natl. Acad. Sci. U.S.A.* 95, 5323–5328. doi: 10.1073/pnas.95.9.5323

Masquelier, T., and Deco, G. (2013). Network bursting dynamics in excitatory cortical neuron cultures results from the combination of different adaptive mechanism. *PloS ONE* 8:e75824. doi: 10.1371/journal.pone.0075824

Mayr, C., Ehrlich, M., Henker, S., Wendt, K., and Schüffny, R. (2007). Mapping complex, large-scale spiking networks on neural VLSI. *Int. J. Appl. Sci. Eng. Technol.* 4, 37–42.

Mayr, C., Noack, M., Partzsch, J., and Schüffny, R. (2010). "Replicating experimental spike and rate based neural learning in CMOS," in *IEEE International Symposium on Circuits and Systems ISCAS 2010* (Paris), 105–108.

Mayr, C., and Partzsch, J. (2010). Rate and pulse based plasticity governed by local synaptic state variables. *Front. Synaptic Neurosci.* 2:33. doi: 10.3389/fnsyn.2010.00033

Mayr, C., Partzsch, J., Noack, M., Hänzsche, S., Scholze, S., Höppner, S., et al. (in press). A biological real time neuromorphic system in 28 nm CMOS using low leakage switched capacitor circuits. *IEEE Trans. Biomed. Circ. Syst.*

Mayr, C., Partzsch, J., Noack, M., and Schüffny, R. (2014). Configurable analog-digital conversion using the neural engineering framework. *Front. Neurosci.* 8:201. doi: 10.3389/fnins.2014.00201

Mayr, C., Partzsch, J., and Schüffny, R. (2009). Transient responses of activity-dependent synapses to modulated pulse trains. *Elsevier Neurocomput.* 73, 99–105. doi: 10.1016/j.neucom.2009.02.019

Mayr, C., Stärke, P., Partzsch, J., Cederstroem, L., Schüffny, R., Shuai, Y., et al. (2012). "Waveform driven plasticity in BiFeO3 memristive devices: model and implementation," in *Advances in Neural Information Processing Systems 25* (Nevada), 1700–1708.

Mejias, J., and Torres, J. (2009). Maximum memory capacity on neural networks with short-term synaptic depression and facilitation. *Neural Comput.* 21, 851–871. doi: 10.1162/neco.2008.02-08-719

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., and Modh, D. S. (2011). "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," in *Proceedings IEEE CICC* (San Jose, CA), 1–4.

Mitra, S., Fusi, S., and Indiveri, G. (2006). "A VLSI spike-driven dynamic synapse which learns only when necessary," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings 2006 IEEE International Symposium on* (Kos: IEEE), 4.

Moradi, S., and Indiveri, G. (2013). An event-based neural network architecture with an asynchronous programmable synaptic memory. *TBioCAS* 8, 1–10. doi: 10.1109/TBCAS.2013.2255873

Noack, M., Mayr, C., Partzsch, J., and Schüffny, R. (2011). "Synapse dynamics in CMOS derived from a model of neurotransmitter release," in *20th European Conference on Circuit Theory and Design ECCTD2011* (Linkoping), 198–201.

Noack, M., Mayr, C., Partzsch, J., Schultz, M., and Schüffny, R. (2012). "A Switched-capacitor implementation of short-term synaptic dynamics," in *Proceedings MIXDES* (Warsaw), 214–218.

Noack, M., Partzsch, J., Mayr, C., and Schüffny, R. (2010). "Biology-derived synaptic dynamics and optimized system architecture for neuromorphic hardware," in *17th International Conference on Mixed Design of Integrated Circuits and Systems MIXDES 2010* (Warsaw), 219–224.

Park, J., Ha, S., Yu, T., Neftci, E., and Cauwenberghs, G. (2014). "A 65k-neuron 73-Mevents/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *IEEE Biomedical Circuits and Systems Conference (BioCAS 2014)* (Lausanne), 675–678.

Rolls, E., and Deco, G. (2010). *The Noisy Brain: Stochastic Dynamics as a Principle of Brain Function*. Oxford, UK: Oxford University Press.

Rolls, E. T., Dempere-Marco, L., and Deco, G. (2013). Holding multiple items in short term memory: a neural mechanism. *PloS ONE* 8:e61078. doi: 10.1371/journal.pone.0061078

Rovere, G., Ning, Q., Bartolozzi, C., and Indiveri, G. (2014). "Ultra low leakage synaptic scaling circuits for implementing homeostatic plasticity in neuromorphic architectures," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on* (Melbourne), 2073–2076.

Roy, K., Mukhopadhyay, S., and Mahmoodi-Meimand, H. (2003). Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proc. IEEE* 91, 305–327. doi: 10.1109/JPROC.2002.808156

Rubinov, M., Sporns, O., Thivierge, J.-P., and Breakspear, M. (2011). Neurobiologically realistic determinants of self-organized criticality in networks of spiking neurons. *PLoS Comput. Biol.* 7:e1002038. doi: 10.1371/journal.pcbi.1002038

Schemmel, J., Bruderle, D., Grubl, A., Hock, M., Meier, K., and Millner, S. (2010). "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on* (Paris: IEEE), 1947–1950.

Scholze, S., Eisenreich, H., Höppner, S., Ellguth, G., Henker, S., Ander, M., et al. (2011). A 32 GBit/s communication SoC for a waferscale neuromorphic system. *Integr. VLSI J.* 45, 61–75. doi: 10.1016/j.vlsi.2011.05.003

Seo, J-S., Brezzo, B., Liu, Y., Parker, B. D., Esser, S. K., Montoye, R. K., et al. (2011). "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proceedings IEEE CICC* (San Jose, CA), 1–4.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodríguez, F. et al. (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. *Neural Netw. IEEE Trans.* 20, 1417–1438. doi: 10.1109/TNN.2009.2023653

Shuai, Y., Ou, X., Luo, W., Du, N., Wu, C., Zhang, W., et al. (2013). Nonvolatile multilevel resistive switching in Ar+ irradiated BiFeO3 thin films. *IEEE Electron Device Lett.* 34, 54–56. doi: 10.1109/LED.2012.2227666

Song, W.-C., Choi, H.-W., Kwak, S.-U., and Song, B.-S. (1995). A 10-b 20-msample/s low-power cmos adc. *Solid State Circ. IEEE J.* 30, 514–521.

Vogelstein, R. J., Mallik, U., Vogelstein, J. T., and Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw.* 18, 253–265. doi: 10.1109/TNN.2006.883007

Yang, M., Liu, S.-C., Li, C., and Delbruck, T. (2012). "Addressable current reference array with 170dB dynamic range," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on* (Seoul: IEEE), 3110–3113.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 30 September 2014; accepted: 09 January 2015; published online: 02 February 2015.

Citation: Noack M, Partzsch J, Mayr CG, Hänzsche S, Scholze S, Höppner S, Ellguth G and Schüffny R (2015) Switched-capacitor realization of presynaptic short-term-plasticity and stop-learning synapses in 28 nm CMOS. Front. Neurosci. ***9****:10. doi: 10.3389/fnins.2015.00010*

This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.

A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks

Runchun M. Wang, **Tara J. Hamilton**, **Jonathan C. Tapson** and **André van Schaik***

The MARCS Institute, University of Western Sydney, Sydney, NSW, Australia

We present a neuromorphic implementation of multiple synaptic plasticity learning rules, which include both Spike Timing Dependent Plasticity (STDP) and Spike Timing Dependent Delay Plasticity (STDDP). We present a fully digital implementation as well as a mixed-signal implementation, both of which use a novel dynamic-assignment time-multiplexing approach and support up to 2^{26} (64M) synaptic plasticity elements. Rather than implementing dedicated synapses for particular types of synaptic plasticity, we implemented a more generic synaptic plasticity adaptor array that is separate from the neurons in the neural network. Each adaptor performs synaptic plasticity according to the arrival times of the pre- and post-synaptic spikes assigned to it, and sends out a weighted or delayed pre-synaptic spike to the post-synaptic neuron in the neural network. This strategy provides great flexibility for building complex large-scale neural networks, as a neural network can be configured for multiple synaptic plasticity rules without changing its structure. We validate the proposed neuromorphic implementations with measurement results and illustrate that the circuits are capable of performing both STDP and STDDP. We argue that it is practical to scale the work presented here up to 2^{36} (64G) synaptic adaptors on a current high-end FPGA platform.

Keywords: mixed-signal implementation, synaptic plasticity, STDP, STDDP, analog VLSI, time-multiplexing, dynamic-assigning, neuromorphic engineering

OPEN ACCESS

Edited by:
Elisabetta Chicca,
University of Bielefeld, Germany

Reviewed by:
Johannes Schemmel,
University of Heidelberg, Germany
Michael Pfeiffer,
University of Zurich and ETH Zurich, Switzerland
Francesco Tenore,
Johns Hopkins University Applied Physics Laboratory, USA

****Correspondence:***
Runchun M. Wang,
The MARCS Institute, University of Western Sydney, Room 109, Building XB, Kingswood Campus, Sydney, NSW 2747, Australia
mark.wang@uws.edu.au

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *21 October 2014*
Accepted: *06 May 2015*
Published: *20 May 2015*

Citation:
Wang RM, Hamilton TJ, Tapson JC and van Schaik A (2015) A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks. Front. Neurosci. 9:180. doi: 10.3389/fnins.2015.00180

Introduction

Plastic synapses, i.e., synapses that can adapt their gain according to one or more adaptation rules, are extremely important in neural systems, as it is generally accepted that learning in the brain arises from synaptic modifications. The Spike Timing Dependent Plasticity (STDP) algorithm (Gerstner et al., 1996; Magee, 1997; Markram et al., 1997; Bi and Poo, 1998), which is one of the adaptation rules observed in biology, modulates the weight of a synapse based on the relative timing between the pre-synaptic spike and the post-synaptic spike. Besides weight adaptation, some observations suggest that the propagation delays of neural spikes, as they are transmitted from one neuron to another, may be adaptive (Stanford, 1987). Axonal delays are an important feature that seems to play a key role in the formation of neuronal groups and memory (Izhikevich, 2006). In our previous work (Wang et al., 2011b, 2012), a delay adaptation algorithm, Spike Timing Dependent Delay Plasticity (STDDP), inspired by STDP was developed to fine-tune delays that had been programmed into the network. We recently showed that the time delays of neural spike propagation in the rat somatosensory cortex can be modified by suprathreshold synaptic processes such as STDP (Buskila et al., 2013). This suggests that it is likely that synaptic weights and the propagation delays are adapted simultaneously.

The main goal of this work is to develop a design framework that is capable of implementing neural networks with maximum size, using simplified biological models. To allow for future implementations that interface with the real world, these neural networks should be running in real time. While detailed simulations of small networks of neurons are one way of studying neural systems, such small networks are not able to capture all the complexity and dynamics of a large scale neural network with non-linear properties, such as a model of neocortex, as pointed out by Johansson and Lansner (2007). In the work reported here, we have therefore focussed on attaining maximum network size.

As synaptic plasticity has not yet been fully characterized and models of synaptic plasticity remain in flux (Brenner and Sejnowski, 2011; Sejnowski, 2012), dedicated hardware implementations that have been hardwired to one particular type of plasticity rule will not be able to adapt to likely future changes in plasticity models. Thus, the design framework we present here will be capable of including various substantial neural networks, each of which may be designed to solve a particular task.

In this paper, we will focus on exploring hardware friendly implementations rather than comparing our learning rules with the vast, well established complex algorithms used in computational neuroscience. As a result of our hardware focus, mathematical analysis of the long-term behavior of the plasticity rules in benchmark networks and quantifying the effects of our learning rules on the synaptic weight, which are commonly used in computational modeling papers, are out of the scope of this paper and will therefore not be addressed.

Simulating neural networks on computers has been successful in informing the computational neuroscience community on promising learning strategies, network configurations and neural models for many decades. This approach, however, does not scale very well, slowing down considerably for large networks with large numbers of variables. For instance, the Blue Gene rack, a two-million-dollar, 2048-processor supercomputer, takes 1 h and 20 min to simulate 1 s of neural activity in 8 million integrate-and-fire neurons (Izhikevich, 2003) connected by 4 billion static synapses (Wittie and Memelli, 2010). For smaller scale networks, Graphic Processing Units (GPUs) can perform certain types of simulations tens of times faster than a PC (Shi et al., 2015). GPUs still perform numeric simulations, however, and, depending on the complexity of the network, it can take hours to simulate 1 s of activity in a tiny piece of cortex (Izhikevich and Edelman, 2008). Along with general hardware solutions, there have been a number of more dedicated hardware solutions (Pfeil et al., 2012, 2013; Painkras et al., 2013). A good example of a dedicated solution that implements numeric simulation of neurons is the SpiNNaker project (Galluppi et al., 2012). In SpiNNaker, ARM processors run software neuron models. Their most recent work shows that the SpiNNaker cores are capable of implementing 96,000 synapses (7500 synapses per core) for STDP in real time (Galluppi et al., 2014).

An alternative approach is to use the analog VLSI (aVLSI) circuits, which avoid any need to discretise differential equations of neuronal dynamics. These implementations will also add stochasticity to the system through electronic noise and device mismatch, resulting in more realistic simulations of biological neural networks. The basic STDP learning rule, which is a paired pulse protocol (Gerstner et al., 1996), has been successfully implemented using aVLSI circuits (Bofill-i-petit and Murray, 2004; Indiveri et al., 2006; Häfliger, 2007; Koickal et al., 2007). More variants of the STDP algorithm have been proposed by Brader et al. (2007a) and Graupner and Brunel (2012). These algorithms capture more of the synaptic dynamics but still follow the principle that the modification of the synaptic weight depends on the relative timing of individual pre- and post-synaptic spikes. Many aVLSI implementations of these algorithms have been proposed (Chicca et al., 2003; Mitra et al., 2009; Giulioni et al., 2012). Similarly, aVLSI circuits have also been used to implement the STDDP learning rule (Wang et al., 2011a,b, 2012, 2013a). This aVLSI approach is useful for studying the dynamics of small and densely interconnected networks, but less so for the study of large and sparsely connected networks, such as complex models of various areas of cortex. The aVLSI implementations all used dedicated synapses for a specific type of synaptic plasticity and the number of plastic synapses integrated on single chip is usually fewer than tens of thousands. This significantly limits the size of the network these approaches can implement.

We chose to implement a synaptic plasticity adaptor array that is separate from the neurons (see **Figure 1**). In this scheme, the address of the pre-synaptic spike from the pre-synaptic neuron will have already been remapped to the address of the post-synaptic neuron by the router shown in **Figure 1**. For each synapse, which remains part of the neuron, a synaptic adaptor will be connected to it when it needs to apply a certain synaptic plasticity rule. The synaptic adaptor will carry out the weight/delay adaptation by updating weight/delay values that are stored in digital memory. For each incoming pre-synaptic spike, the adaptor will send a weighted/delayed pre-synaptic spike to the post-synaptic neuron in the neuron array.

This strategy provides great flexibility, as a hardware neural network can be configured to perform multiple synaptic plasticity rules without needing to change its own structure, simply by connecting the synapses to the appropriate modules in the synaptic plasticity adaptor array. This structure was first proposed by Vogelstein and his colleagues in the IFAT project (Vogelstein et al., 2007). However, they didn't implement synaptic plasticity in that work although they did discuss the implementation of STDP with this structure. It seems that this flexibility will generate a communication overhead. The communication between neurons and adaptors has the same overhead as the communication between neurons and other neurons in a network without a separate adaptor array. Thus, the additional overhead stems from the communication from the adaptor array to each of the synapses. This will be discussed in more detail in the next section. The major disadvantage of our approach is that it is incapable of modeling the ion channels in the biological synapses. Compared to the aVLSI approach, our approach is less useful for studying the

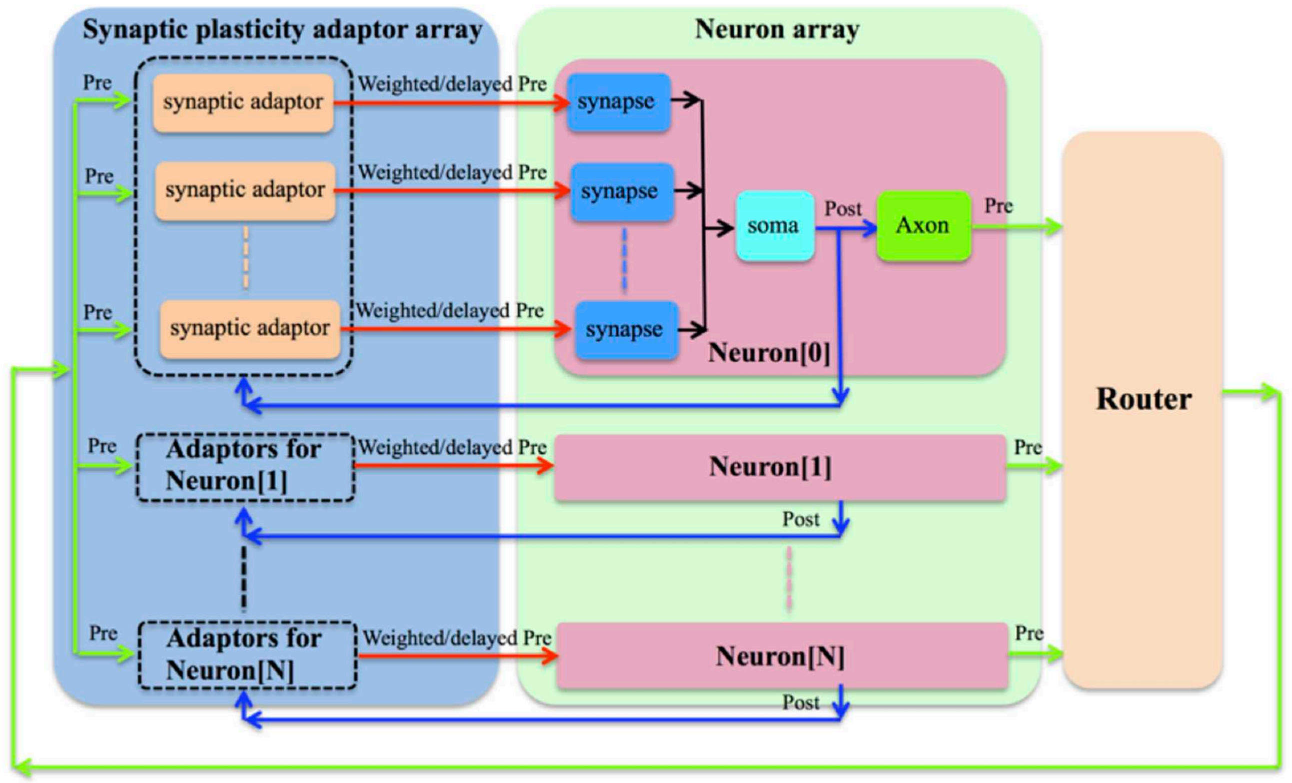


FIGURE 1 | The synaptic plasticity adaptor array that is separate from the neurons. For each synapse, which remains part of the neuron, a synaptic adaptor will be connected to it when it needs to apply a certain synaptic plasticity rule. The synaptic adaptor will carry out the weight/delay adaptation by updating weight/delay values. For each incoming pre-synaptic spike, the adaptor will send a weighted/delayed pre-synaptic spike to the post-synaptic neuron in the neuron array.

dynamics of the networks that require high degrees of biological realism. Furthermore, our approach is less power efficient compared to the aVLSI implementations. This is because our implementation has to employ configurable but power hungry devices, such as FPGAs/MCUs, to achieve its flexibility. Analog VLSI implementations, in contrast, especially those operating in weak inversion (Liu et al., 2002), are capable of achieving a significantly low power consumption.

We have previously presented a compact reconfigurable mixed-signal implementation of a synaptic plasticity adaptor that is capable of performing both STDP and STDDP (Wang et al., 2014a). Here, we present its follow-up work that uses a novel approach to scale up the numbers of synaptic plasticity adaptors up by 128 (2^7) times more without increasing the hardware cost significantly. While the design of the router and the neuron arrays are out of the scope of this paper and will not be presented.

Materials and Methods

Learning Rules

Spike Timing Dependent Plasticity

The STDP algorithm modulates the weight of a synapse based on the relative timing of the pre- and post-synaptic spikes. The weight of a synapse will be increased if a pre-synaptic spike arrives several milliseconds before the post-synaptic spike fires. Conversely, the weight will be decreased in the case that the post-synaptic spike fires earlier than the arrival of a pre-synaptic spike by several milliseconds. The amount and direction of modification of the weight are determined by the time between the arrival of the pre- and post-synaptic spike.

To obtain this time difference, we need to know when the pre- and post-synaptic spike arrives. This is implemented by introducing a time window generator, which is composed of a 4-bit counter, it will be reset by either spike and increased by one bit at each time step, e.g., 1 ms until it reaches its maximum value 0 × F. The time at which the alternative spike arrives is represented by the value of the counter. We also define that the time window is "active" before it reaches its maximum value. As we assume that the adaption will not be carried out if the pre- and post- synaptic spikes arrive simultaneously, only one time window generator will be needed.

In the original STDP learning rule (Gerstner et al., 1996), the amount of synaptic modification is summarized by the following equations:

$$\Delta w = \begin{cases} A^{+} exp\left(\Delta t/\tau_{+}\right), & if\ \Delta t < 0 \\ -A^{-} exp\left(\Delta t/\tau_{-}\right), & if\ \Delta t \geq 0 \end{cases} \tag{1}$$

where Δw is the modification of the synaptic weight, Δt is the time difference between the arrival time of the pre-

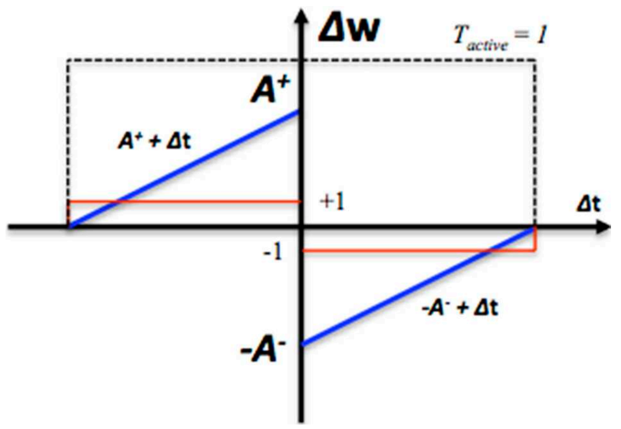


FIGURE 2 | The STDP modification function. Δ*t* is the time difference between the arrival time of the pre- and post-synaptic spike. The blue line represents synaptic modification Δ*w*, which is linearly proportional Δ*t*. The red line represents the synaptic modification Δ*w*, which is a fixed step. The dashed line represents the range of pre-to-post-synaptic interspike intervals over which synaptic modification is performed.

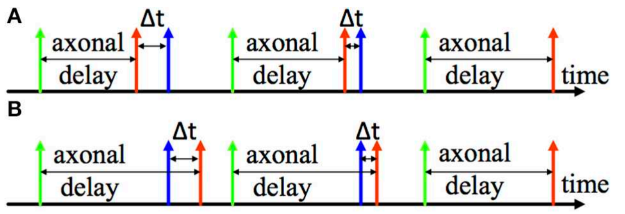


FIGURE 3 | Illustration of Delay adaptation. (A) Delay increment; **(B)** Delay decrement. The axonal delay presents the delay between the firing time of the pre-synaptic neuron (the green one) and arrival time of the pre-synaptic spike (the red one) at the post-synaptic neuron. Δ*t* represents the time difference between the pre- and post-synaptic spike (the blue spike) to and from the post-synaptic neuron.

and post-synaptic spike. The maximum amounts of synaptic modification Δw are determined by two positive parameters: A^+ and A^-. The ranges of pre-to-post-synaptic interspike intervals over which synaptic modifications are performed are determined by the parameters τ_+ and τ_-. The authors in Song et al. (2000) concluded that this function provides a reasonable approximation of the dependence of synaptic modification on spike timing observed experimentally. However, it is a computationally intensive function since it requires exponentiation and division operations, both of which would occupy a large silicon area.

To reduce the required silicon area, in our system, we have implemented two simplified modification rules. The first one is to change the weight proportionally to the calculated time difference (see the blue line in **Figure 2**) and is summarized by the following equations:

$$\Delta w = \begin{cases} A^+ + \Delta t, \ if \ T_{active} = 1 \ and \ \Delta t < 0 \\ -A^- + \Delta t, \ if \ T_{active} = 1 \ and \ \Delta t > 0 \end{cases} \tag{2}$$

where Δw is the modification of the synaptic weight, Δt is the time difference between the arrival time of the pre- and post-synaptic spike. T_{active} is a Boolean value that indicates the time window generator is active (see the dashed line in **Figure 2**). In this system, the synaptic weight is an unsigned integer, which ranges from 0 to 15. A^+ and A^- are both set to 16 here. The second one is to change the value of the weight by a fixed value (see the red line in **Figure 2**) and is summarized by the following equations:

$$\Delta w = \begin{cases} + step, \ if \ T_{active} = 1 \text{ and } \Delta t < 0 \\ - step, \ if \ T_{active} = 1 \text{ and } \Delta t > 0 \end{cases} \tag{3}$$

where *step* is the fixed value and is set to 1 here. No weight modification will be performed if the pre- and post-synaptic spikes arrive simultaneously. The efficacy of these two simplified learning rules will be presented in Section Performance of STDP.

Spike Timing Dependent Delay Plasticity

Two examples of the adaptation of axonal delays are shown in **Figure 3**, an increment of the delay (**Figure 3A**) and a decrement of the delay (**Figure 3B**). After the pre-synaptic neuron fires there is an axonal delay before the delayed pre-synaptic spike is sent to the post-synaptic neuron. If the post-synaptic spike, which is from the post-synaptic neuron, is not simultaneous with the delayed pre-synaptic spike, we may adapt the axonal delay by increasing or decreasing it by a small amount. This procedure is repeated until the delayed pre-synaptic spike occurs simultaneously with the post-synaptic spike.

Since this learning rule also needs to obtain the time difference between the pre- and post-synaptic spikes, we will use the same time window generator as described above, to generate the axonal delay. In this case, however, the time window generator will be started by the pre-synaptic spike (the green spike in **Figure 3**). Moreover, the duration of the generated time window will be modulated according to the axonal delay. The modification of the axonal delay will only be performed by the post-synaptic spike: when the post-synaptic spike arrives, if the time window is active, then there is a decrease the axonal delay and vice versa. The modification of the axonal delay Δd is summarized by the following equations:

$$\Delta d = \begin{cases} - step, \ if \ T_{active} = 1 \\ + step, \ if \ T_{active} = 0 \end{cases} \tag{4}$$

where *step* is a fixed value and is set to 1 here. Modifying the axonal delay by a single step is one of the three strategies, which were proposed and proved to be functional in our previous work (Wang et al., 2013b). No delay modification will be performed if the delayed pre-synaptic spike and post-synaptic arrive simultaneously. In this system, the axonal delay is also an unsigned integer, which ranges from 0 to 15.

Design Choice

To implement multiple synaptic plasticity rules for large scale spiking neural networks, the design choice we made were based on the following principles:

Time-multiplexing

In digital implementations of spiking neural networks, a single physical neuron can be time-multiplexed to simulate many virtual neurons, since digital hardware neurons can operate much faster than biological neurons. Each virtual neuron only needs to be updated every millisecond or so, as a millisecond time resolution is generally acceptable for neural simulations. Digital implementations of neurons using this time-multiplexing approach have been described in Cassidy and Andreou (2008); Cassidy et al. (2011); Wang et al. (2013b, 2014c). In the implementation presented here, we are time-multiplexing both the synaptic adaptors and the neurons.

Dynamic-assignment

It is not necessary to implement all neurons physically on silicon as based on the physiological metabolic cost of neural activity, it has been concluded that fewer than 1% of neurons are active in the brain at any moment (Lennie, 2003). A larger address space can be mapped onto a smaller number of physical components through dynamically assigning these components. Based on this principle, we have presented a dynamically-assigned digital and analog neuron array in Wang et al. (2013b) and Wang et al. (2014d), respectively. In these two systems, 4096 (4 k) neurons were achieved with only tens of neurons implemented physically on silicon. Here we also use this approach for both the neurons and the synaptic adaptors.

Mixed-signal

This implementation style can combine some of the advantages of both analog and digital implementations. Analog implementations can realize biological behaviors of neurons in a very efficient manner, whereas digital implementations can provide the re-configurability needed for rapid prototyping of spiking neural networks. As a result, mixed-signal implementations offer an attractive solution for implementing neural networks and many designs have been proposed for such systems (Goldberg et al., 2001; Gao and Hammerstrom, 2007; Mirhassani et al., 2007; Vogelstein et al., 2007; Harkin et al., 2008, 2009; Schemmel et al., 2008; Saighi et al., 2010; Yu and Cauwenberghs, 2010; Zaveri and Hammerstrom, 2011; Minkovich et al., 2012).

Standardization

To enable multiplexing building blocks, such as neurons, synapses, and axons, in a neuromorphic system, these circuits must be designed as standardized building blocks with a standard protocol for communication with programmable devices. Specifically for use in time-multiplexed neural systems, we have developed a synchronous Address Event Representation (AER) protocol, which uses a collision-free serial processing scheme with a single active signal and an address (Wang et al., 2013b). This synchronous scheme eliminates the overhead of an arbiter in the standard AER protocol.

For the maximum utilization of a fixed sized aVLSI chip, it is best to reduce the on-chip routing as much as possible as the routing can be carried out off-chip by FPGAs or microprocessors with more flexibility and extensibility. As the on-chip topology of the aVLSI circuits is generally fixed after fabrication, it is better to implement the whole system in an FPGA for prototyping and optimization before fabricating the aVLSI chips.

Pulse width Modulation

For the systems that are sensitive to high communications overheads, e.g., aVLSI chips with limited number of pads, we adopted a pulse width modulation scheme, to minimize the communication bandwidth. In this scheme the durations of the spikes are modulated according to the synaptic weights, and the synapses in the neuron array are sensitive to the durations of the spikes (e.g., Wang et al., 2014b). It should be noted, however, that we could easily reconfigure the system to send out synaptic weights directly to the neurons in systems that are not sensitive to high communications overheads, e.g., FPGA designs.

Versatility

To efficiently implement synaptic plasticity in large-scale spiking neural networks with different learning rules, the building block should be capable of being configured for multiple synaptic plasticity rules, such as STDP and STDDP. When the synaptic plasticity adaptor is configured as the STDP adaptor, it performs STDP by receiving pre- and post-synaptic spikes from the pre- and post-synaptic neuron respectively. Its output, a weighted pre-synaptic spike generated using pulse width modulation, is sent to the synapse of the post-synaptic neuron for generating a post-synaptic current (PSC). When the synaptic plasticity adaptor is configured as an STDDP adaptor, it receives the same signals, but its output is a pre-synaptic spike that has been delayed according to the stored delay value for this neuron-to-neuron connection.

Architecture

Figure 4 shows the topology of the proposed mixed-signal synaptic plasticity adaptor array. It consists of an adaptor array on an FPGA and a time window generator array, which could be either a fully digital implementation on the same FPGA, or an analog implementation on a custom designed aVLSI chip, or both, as shown. All blocks use time multiplexing and are dynamically assigned using an FPGA to control the assignment.

Based on the physiological metabolic cost of neural activity, it has been concluded that fewer than 1% of neurons are active in the brain at any moment (Lennie, 2003). The anatomical studies of neocortex presented in Scannell et al. (1995) showed that cortical neurons are not randomly wired together. Instead, cortical neurons are typically organized into local clusters called minicolumns, which are then grouped into modules called hypercolumns (Hubel and Wiesel, 1974; Amirikian and Georgopoulos, 2003). The connections of the minicolumns are highly localized so that connectivity between two nearby (less than 25–50 μm apart) pyramidal neurons is high and the connectivity between two neurons drops sharply with distance (Holmgren et al., 2003). Based on the experimental data in Tsunoda et al. (2001) and Johansson and Lansner (2007) concluded that *at most a few percent of the hypercolumns and hence only about 0.01% of the minicolumns and neurons are active in a functional sense (integrating and firing) at any moment in the cortex.* They also concluded that only 0.01% of the synapses

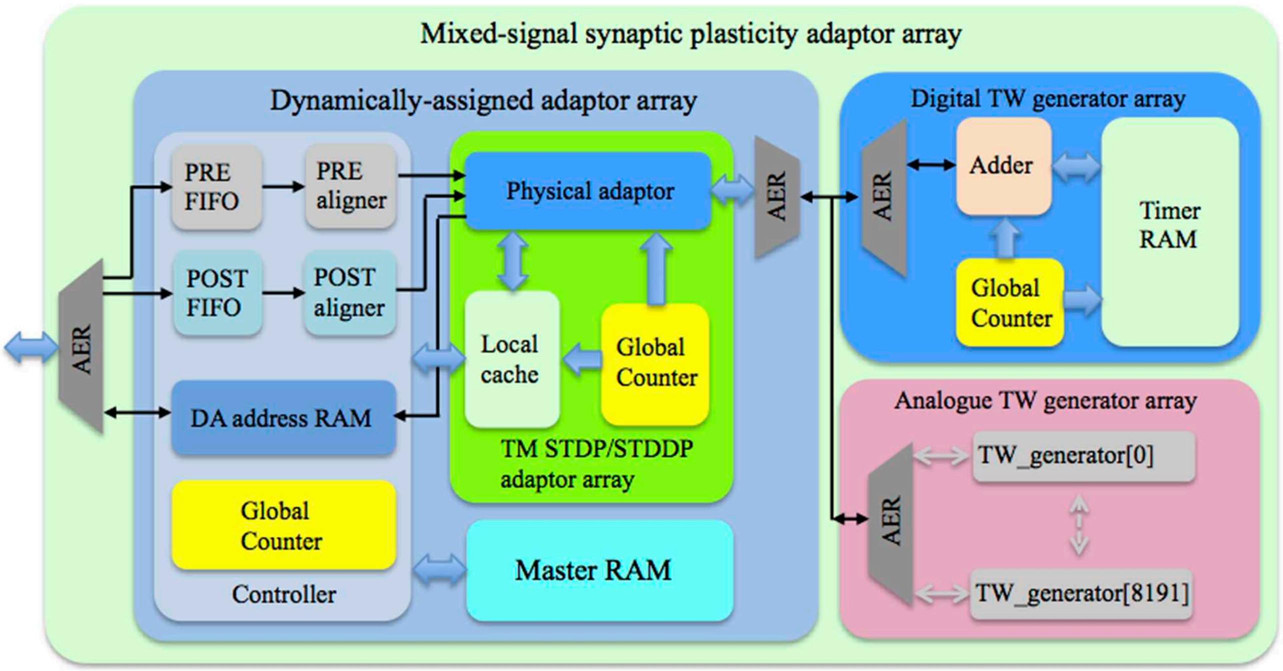


FIGURE 4 | Topology of the mixed-signal synaptic plasticity module array. The controller receives pre- and post-synaptic spikes from the neuron array and assigns them to the corresponding TM adaptors according to their addresses. The global counter processes each TM adaptor sequentially. We use the Master RAM to store all the weight/delay values, while the TM STDP/STDDP adaptor array has a Local cache that stores the values of the DA adaptors that are being processed. The time window generator array generates the time windows that will be used by the TM adaptors for performing the learning rules.

in our brains are active (transmitting signals) on average at any moment. Hence, in principle, one hardware synapse could be dynamically reassigned to 10^4 virtual synapses on average. Such a hardware synapse will be referred to as a physical synapse and the synapse to be simulated will be referred to as a dynamically-assigned (DA) synapse. If a DA synapse cannot be simulated in a single time step, the physical synapse needs to be assigned to that DA synapse for a longer time and the number of DA synapses a single physical synapse can simulate will go down proportionally.

On an FPGA running at 200 MHz, we can time-multiplex a single physical synapse to simulate $1\,\text{ms}/5\,\text{ns} = 200{,}000$ time-multiplexed (TM) synapses, each one updated every millisecond. Therefore, theoretically, a TM synapse array with 200,000 TM synapses can be dynamically assigned for $200{,}000 \times 10^4 = 2 \times 10^9$ DA synapses, if these synapses can be simulated in a single 5 ns clock cycle and if only 0.01% of the synapses are active at any time step.

Since we chose to implement a synaptic plasticity adaptor array that is separate from the neurons, we will apply these two approaches to the adaptors. To be able to deal with higher synaptic activity rates, and because powers of two are preferable to optimize memory use for storing variables, such as weights and delays, we chose to dynamically assign one TM adaptor for 8192 (8 k) DA adaptors. The maximum active rate of the synapses that this system can support is therefore $1/8\,\text{k} \approx 0.012\%$. The TM adaptor array itself is configured to simulate 8 k TM adaptors, allowing it to support $8\,\text{k} \times 8\,\text{k} = 64\,\text{M}$ synapses. Each TM adaptor can use up to 25 clock cycles to complete its processing to maintain an update rate of 1 kHz (the corresponding time step is about 1 ms). The time window generator array is also configured to have 8 k identical time window generators, each time window being assigned to one TM adaptor.

The dynamically-assigned adaptor array consists of three sub-blocks: a controller, a TM STDP/STDDP adaptor array and a Master RAM. A single physically implemented dynamically-assigned adaptor array is capable of representing up to 64M DA adaptors, thus the hardware cost of the DA adaptors is negligible. The physical constraint for this approach is data storage. On-chip SRAM (on an FPGA) will be highly limited in size (generally less than tens of MBs), while the use of off-chip memory will be limited by the communications bandwidth. It is difficult, but not impossible, to use off-chip memory with the time-multiplexing approach, as new values need to be available from memory every time slot to provide real-time simulation.

Since we are aiming for the maximum network size, we need to ensure that the system is able to utilize off-chip memory. Inspired by the cache structure used in state-of-the-art CPUs, we use the Master RAM to store all the weight/delay values, while the TM adaptor array has a Local cache that stores the values of the DA adaptors that are being processed. The accessing (read/write) of the Master RAM will only be performed when needed. This means that new values are no longer required to be available from memory every time slot. Hence this cache structure greatly reduces the bandwidth requirement to use external memory.

We will present the details of this cache structure following a presentation on the management the incoming spikes. It should be noted, however, that using off-chip memory requires flow control for the memory interface, which results in a more complex system architecture. Thus, for the work reported here, we use only on-chip memory, thus simplifying the system architecture. We will discuss the usage of off-chip memory in more detail in Section Discussion.

The controller receives pre- and post-synaptic spikes from the neuron array (see **Figure 4**) and assigns them to the corresponding TM adaptors according to their addresses. In our previous work (Wang et al., 2013b, 2014d) that also implemented the dynamic-assignment algorithm, the controller needs to check whether there is already a neuron assigned to the incoming spike or not. This method has a high usage of slice LUTs, which is the bottleneck for large-scale FPGA designs. This is because that method requires an address register array and a timer array, both of which are running in parallel and hence have to be implemented with slice LUTs.

To avoid this problem, we chose instead to use a direct mapping method that assigns one fixed TM adaptor as the target adaptor for the incoming spike irrespective of whether the TM adaptor has been assigned or not. The incoming spike's AER address is a 26-bit address (along with a single active line). We only store the most significant 13 bits out of 26 bits into a DA address RAM (a dual port RAM with a size of 8 k × 13 bits), while the other 13 bits determine where, i.e., in which position of the DA address RAM, the 13 bits will be stored.

To decouple writing new events (pre- and post-synaptic spikes) from reading out from current events, we use a FIFO and an aligner (a dual port RAM with a size of 8 k × 1 bit that corresponds to 8 k TM adaptors). For pre- and post-synaptic spikes, the size of the FIFO is 16 × 26 bit and 16 × 13 bit respectively. The work presented in Cassidy et al. (2011) used two banks of dual port RAM to implement a ping-pong buffer. This requires much more RAM than our solution.

Figure 5 shows the timing diagram for the controller for one time slot. Assuming the PRE_FIFO is empty at T0, when a new pre-synaptic spike arrives (its active line is high) at T2, its 26-bit AER address will be written into PRE_FIFO. The controller will then read the PRE_FIFO by asserting fifo_rd at T3 (since the PRE_FIFO is not empty anymore) and read data (fifo_rddata) will be ready at T4 (one clock cycle latency). To indicate that a spike has arrived (for that TM adaptor), at T4, the controller will write 0 × 1 into the PRE_aligner to the position determined by the least significant 13 bits of the fifo_rddata.

At T4, the controller will also use the least significant 13 bits of fifo_rddata to retrieve the stored address (from the DA address RAM), which will be ready at T6 (two clock cycles latency). If this retrieved address does not match the most significant 13 bits of the delayed fifo_rddata (the red one), this indicates that the target DA adaptor is not the one that has been assigned before. Hence the value (in the Local cache) of the TM adaptor needs to be updated with the value of the target DA adaptor. Therefore, at T6, the controller will read the Master RAM by asserting a read enable signal (M_rden) with a read address M_rdaddr, which is the fifo_rddata signal delayed. For the same reason, at T6, the controller will also update the DA address RAM with the address of this newly arrived pre-synaptic spike: the most significant 13 bits of the delayed fifo_rddata (the least significant 13 bits

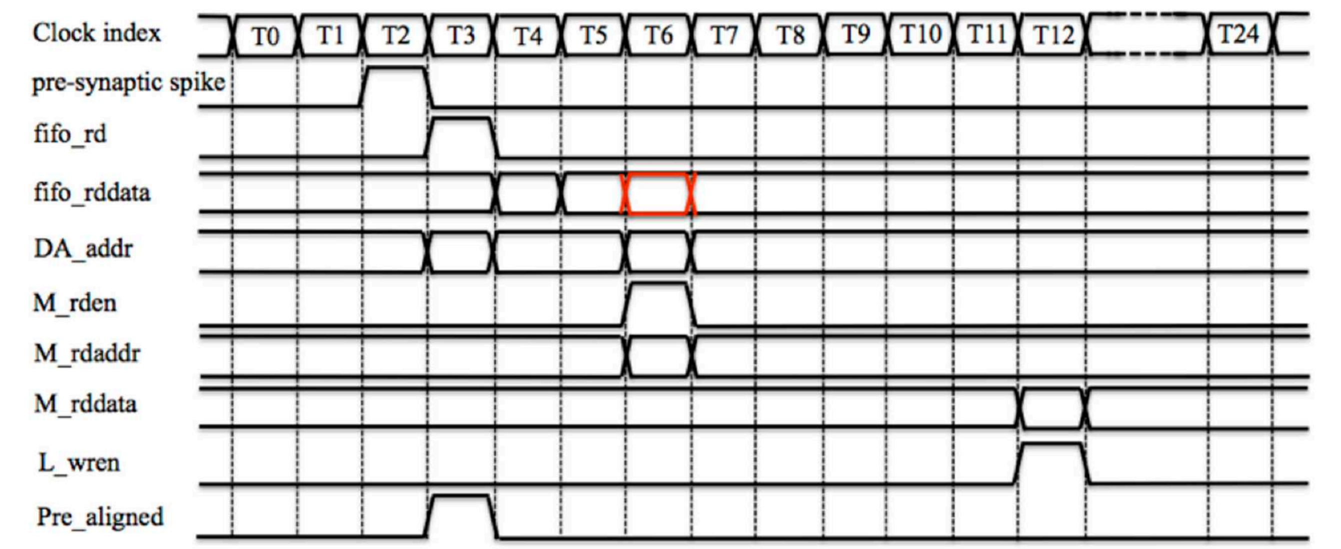


FIGURE 5 | The controller's timing diagram of one time slot. A pre-synaptic spike arrives at the controller at T2 and it will be written into the PRE_FIFO. The controller will read the PRE_FIFO at T3 and the read data (fifo_rddata) will be ready at T4. To indicate that a spike has arrived (for that TM adaptor), at T4, controller will write 0 × 1 into the PRE_aligner to the position determined by the least significant 13 bits of the fifo_rddata. The controller will also use the least significant 13 bits of fifo_rddata to retrieve the stored address (from the DA address RAM), which will be ready at T6. At T6, the controller will read the Master RAM by asserting a read enable signal (M_rden) with a read address M_rdaddr, which is the fifo_rddata signal delayed (the red one). At T12, the output from the Master RAM M_rddata will be ready and the Local cache will be updated by asserting L_wren.

determines the position to write). The output from the Master RAM M_rddata will be ready 6 clock cycles later (we will explain why this latency is needed Section TM STDP Adaptor Array) and the Local cache will be updated by asserting L_wren.

To read out the aligned pre-synaptic spike (Pre_aligned), at each time slot, the controller will read out the TM adaptor of that time slot, from the PRE_aligner at T1. The Pre_aligned will be ready at T3 where the corresponding TM adaptor will acknowledge it, after which it will be cleared. To avoid the collision that can happen when the PRE_FIFO and the controller both try to write the PRE_aligner (at the same location, although this case will not happen frequently) at T3, we set it so that the PRE_FIFO cannot be read (fifo_rd cannot be asserted) at T0. This is indeed the reason to introduce the FIFO since in this way all operations are fully pipelined and can be performed on every clock. To keep consistent with this pipeline, in each time slot, the controller will read out the DA_addr (no clear operation needed) for the TM adaptor of that time slot from the DA address RAM at T1. When the TM adaptor generates a weighted/delayed pre-synaptic spike, the controller will send this spike to the post-synaptic neuron with a 26-bit AER address, which is a combination of the DA_addr and the value of the global counter.

The timing for post-synaptic spike is aligned using a very similar scheme to that described above for the pre-synaptic spike, however, its address will not be stored in the DA address RAM. This is due to the fact that only the weighted/delayed pre-synaptic spike will be sent to the post-synaptic neuron (see **Figure 1**) and hence only the address of the pre-synaptic spike needs be stored and only the pre-synaptic spike will retrieve the weight/delay value from the Master RAM.

This method significantly reduces the usage of the Slice LUTs and hence makes it practical to apply the dynamic-assignment approach to an adaptor array with 8 k neurons. The hardware cost of the FIFO and the aligner is very small and they are both efficiently implemented with on-chip distributed SRAM. It does need a DA address RAM, which needs to be implemented with on-chip block SRAM, but storing only 13 bits significantly reduces the size of the address memory. Another major advantage of this method is its flexibility, e.g., with a 26-bit AER address, input spikes can arrive at any time and be handled. This means multiple different types of neuron arrays can be connected to one adaptor array. Moreover, it suffers little from the large latencies in the spikes, that can be of the order of hundreds microseconds, due to routing. Excessive latency due to routing is quite common in large-scale neural networks. Similarly, the communication overhead between the neuron array and the adaptor array will barely affect the performance of the system.

A collision will happen when multiple input spikes, that target different DA adaptors, while at the same time need the same TM adaptor, arrive within one time step. In this case, only the last arriving spike will be sent to its target adaptor, and the ones that arrived previously will be simply discarded. Another collision will happen when the most significant 13 bits of the address of an incoming post-synaptic spike do not match the DA_addr of the target TM adaptor. In this case the 13 bits of the address of the post-synaptic spike will still be sent to that TM adaptor for performing adaptation and might cause wrong weight/delay modifications. These two possible collision scenarios are drawbacks of our approach, and do affect small and densely interconnected neural networks with high activity rates. These scenarios, however, are not serious problems for large-scale neural networks, the connections of which are highly localized, while the activity rate is low. For instance if we are modeling hypercolumns in human cortex, the experimental data shows that only a few hypercolumns in the human cortex are active for any given task.

For practical applications, within a short period, the TM adaptors should only be assigned for one certain task. When that task ends, they will be released and can then be used by other tasks. For example, one hypercolumn could use all the 8 K TM plastic synapses for learning patterns, which might last for hundreds of milliseconds. After the patterns have been learned (stored in the Master RAM), another hypercolumn could then use these 8 K TM adaptors for learning patterns. It is of course possible that a synapse in another hypercolumn becomes active more or less spontaneously. These spontaneously activated synapses, however, would be uniformly distributed all over the neural network and are thus unlikely to make up a large fraction of the group of synapses in the hyper column that is currently learning patterns. Hence, these spontaneously activated synapses will not have a significant effect on the learning being performed. We will validate the dynamic-assignment scheme in Section Validation of the Dynamic-assignment Scheme. The maximum memory update speed, which is indeed the maximum firing rate of the neurons, that our system supports is 200 Mhz/25 = 8 MHz (much higher than biological neurons).

Time Window Generator Array

The time window generator array has been successfully implemented on a custom designed aVLSI chip, and independently also on the same FPGA as the dynamically-assigned adaptor array. The digital implementation used time-multiplexing to achieve 8 k TM time window generators. However, this fully digital implementation needs block SRAM, as the internal state of each generator needs to be stored in memory in between updates. This memory demand is the real bottleneck of the time-multiplexing approach (Moore et al., 2012). Nevertheless, this fully digital solution will be quite suitable for the applications when aVLSI is not available. On the other hand, an aVLSI circuit can implement a time window generator very efficiently, as long as high precision is not required. Using the aVLSI time window generator circuit reduces memory usage and the memory saved can be used for storing more synaptic weight and delay values, allowing for larger networks. Furthermore, the analog time window generator will add stochasticity to the weight and delay adaptation through electronic noise and device mismatch, which will provide more realistic simulations of biological neural networks.

Analog Time Window Generator Array

We provide a brief review of the analog time window generator, which has been presented in depth in Wang et al. (2014a). **Figure 6A** shows the schematic of the analog time window

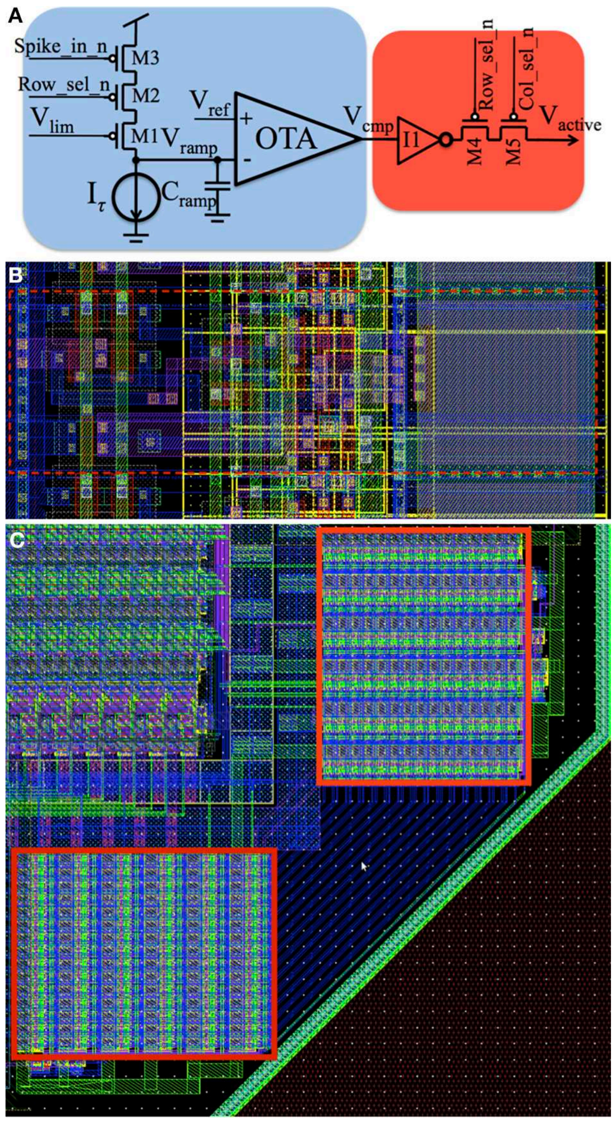


FIGURE 6 | aVLSI time window generator. (A) Schematic; **(B)** Layout; **(C)** Layout of the array. It is placed in a two-dimensional array and when a time window generator is selected, the voltage at node V_{cmp} will pull the output signal of this neuron V_{active} either up to V_{dd} (active, V_{cmp} is low) or down to ground (inactive, V_{cmp} is high) via an inverter (I1) and a serial switch (M4–M5).

generator, comprising a ramp generator circuit (blue) and an AER hand-shaking circuit for our synchronous AER (red). It is placed in a two-dimensional array and therefore requires row and column select signals (*Row_sel_n* and *Col_sel_n*), which are both low when the ramp generator has been selected. When a time window generator is selected, the voltage at node V_{cmp} will pull the output signal of this neuron V_{active} either up to V_{dd} (active, V_{cmp} is low) or down to ground (inactive, V_{cmp} is high) via an inverter (I1) and a serial switch (M4–M5). When this time window generator is not selected (M4 and M5 are OFF), V_{active} will be driven by another other time window generator in the array. Each time window generator is linked to its corresponding TM adaptor and will be processed sequentially, with each generator selected for one time slot. To use the asynchronous aVLSI circuits with the FPGA, synchronization with its clock domain is needed. Since the output signal V_{active} is a 1-bit signal, we use the general method that uses two serially connected flip-flops to sample the input (Weste and Harris, 2005).

This circuit was implemented in the IBM 130 nm technology. For the maximum utilization of silicon area, one time window generator should share as many resources as possible with its neighboring ones. Based on this principle, all the pMOS transistors are located in the right side and all the nMOS transistors are located at the left side (see the dashed red rectangle in **Figure 6B**) so that they can share their bulk connections with each other. All the input/output signals and the bias currents are placed vertically so that they can be merged to a bus across the array without any extra wiring cost. The effective size of a time window generator in the array is $\sim 50\,\mu\text{m}^2$ achieving a density of 20,000 cells/mm^2. As a proof of concept, we have placed 180 of the proposed aVLSI time window generators on the bottom right corner of a test chip, as shown by the red rectangles in **Figure 6C** (Wang et al., 2014b).

Digital Time Window Generator Array

The digital time window generator has the exact same function as the aVLSI time window generator. The global counter processes each TM time window generator sequentially. In each time slot, the controller will read the value of the TM time window generator from the Timer RAM. A counter will be incremented by one at each clock cycle when the digital input spike from the time-multiplexed adaptor is active (high), so that its count increases proportional to the spike width. When there is no input spike, the count will decrease by one each time slot, until it reaches zero, indicating the end of the time window.

Slightly different to the aVLSI time window generator, the output of the digital time window generator contains not only an active line, to indicate whether the time window is finished or not, but also the actual value of the counter. In aVLSI it would be difficult to read out the actual value of the V_{ramp} (see **Figure 6A**) in an efficient manner. In a digital implementation this value is directly accessible to the DA adaptor array and could be used to perform more complex plasticity rules.

TM Adaptor Array

TM STDP Adaptor Array

When implementing the TM adaptor array for STDP, a significant reduction in memory usage was achieved by storing a bistable weight in the Master RAM. This is based on the work by Brader et al. (2007b), which shows that from a theoretical perspective, having only two stable states for synaptic weights does not degrade the performance of associative networks, if the transitions between the stable states are stochastic. For networks with large numbers of neurons, each with large numbers of synapses, the assumptions that synaptic weights will be discretized to two stable values on long time-scales is not too severe, and is supported by biological evidence (Bliss and Collingridge, 1993; Petersen et al., 1998).

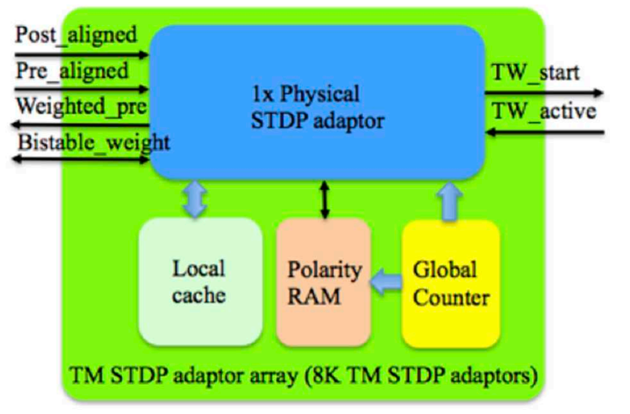


FIGURE 7 | Structure of the TM STDP adaptor array. The global counter processes each TM STDP adaptor sequentially. The Local cache stores the weight values of the TM STDP adaptor that are being processed. The learning rules will be performed with the aligned pre- and post- synaptic spikes from the controller and the active line from the time window generator.

Figure 7 shows the structure of the TM STDP adaptor array, which consists of a physical STDP adaptor, a Local cache, a polarity RAM and a global counter. The Local cache, which is a dual port RAM with a size of 8 k × 4 bit, stores the weights (4-bit resolution) of the TM adaptors. The Master RAM will only need to store one of the two stable values of the bistable weight, and thus needs only 1 bit per weight. When a TM adaptor has been assigned to a DA adaptor, its bistable weight will be read out from the Master RAM. We then use this bistable weight to generate one random 4-bit weight (stored in the Local cache) for that TM adaptor. Only when there is a modification of the 4-bit weight, which will generate a bistable weight simultaneously, will we need to update the Master RAM with this bistable weight.

Since there is only one time window generator per TM adaptor, it will have the wrong weight modifications when multiple spikes (of the same type, e.g., pre-synaptic spikes) arrive within the duration of one time window. For instance, one pre-synaptic spike starts a time window while the pre-synaptic spikes that follow and arrive within this time window will perform weight decrement. To solve this problem, the polarity RAM, which is a dual port RAM with a size of 8 k × 1 bit, was introduced. The polarity RAM stores the polarity of the time window for each TM adaptor. For the time window started by pre- and post-synaptic spikes, the polarity value is 0 × 0 and 0 × 1 respectively. The time window is set such that it will be restarted for each of the multiple spikes received within the time window. In other words, each incoming spike will either start a time window or perform weight modification.

The read out from the Master RAM and the update of the Local cache was presented with the timing diagram of the controller in Section Architecture. Since the retrieved bistable weight from the Master RAM is 1-bit while the weight to be written into the Local cache is 4-bits, this bistable weight will be used as the most significant bit (MSB) of that 4-bit weight. To keep the transitions between the stable states stochastic, the remaining 3 bits are generated pseudo-randomly by a linear feedback shift register (LFSR).

Figure 8 shows the timing diagram for performing the STDP algorithm by one TM STDP adaptor. **Figure 8A** shows how a pre-synaptic spike starts a time window. Pre_aligned is ready at T3 and TW_active will be ready at T8 (comprising 7 clock cycles for latency and 2 clock cycles for synchronization). As the time window is inactive, the delayed Pre_aligned (the red one) will start the time generator at T8 by sending a pulse (TW_start, starts at T9) that controls the duration of the window. Note the duration of the time window is fixed during operation but the parameter is configurable. The polarity of the time window, which is 0 × 0, will be written to the polarity RAM by asserting Pol_wr at T9. Since the time window is inactive, no weight modification is needed and neither the Local cache nor the Master RAM needs to be updated.

Since the incoming spike is a pre-synaptic spike, we need to generate the weighted pre-synaptic spike, which will be sent to the post-synaptic neuron. The local weight (L_rddata) is read out at T0 and ready at T2 (two clock cycles latency), the delayed Pre_aligned will send out the Weighted_pre and assert its active line (Pre_active_line) at T9. Simultaneously, the controller will send this spike to the post-synaptic neuron with a 26-bit AER address (Pre_addr), which is a combination of the delayed DA_addr (the red one) and the value of the global counter.

Figure 8B shows the timing diagram for increasing the synaptic weight. Assuming the time window has already been started by a previous pre-synaptic spike. The polarity of the time window (TW_polarity) and the local weight (L_rddata) are both read out at T0 and ready at T2 (two clock cycles latency). Post_aligned is ready at T3 and TW_active will be ready at T8. Since TW_active is active and TW_polarity is low, which indicates that this time window was started by a pre-synaptic spike, the delayed Post_aligned (the red one) will increase L_rddata by one (when using the aVLSI time window generator array) or by the value of the time window generator's counter (when using the digital time window generator array).

The updated weight (L_wrdata) will be written into the Local cache by asserting L_wren at T9. In the controller, the latency from fifo_rd, which cannot be asserted at T0, to L_wren is 10 clock cycles (see **Figure 5**). Hence a collision when the TM STDP adaptor and the controller are updating the Local cache at the same cycle will never happen. This is why the latency from M_rden to M_rddata is set to 6 clock cycles. The idea behind this setting is to achieve a fully pipelined design so that all operations can be performed on every clock cycle and there are no stalls in the pipeline. Note that if we were using only a digital time window generator array, the time slot could be optimized to less clock cycles by using tens of pipeline stages (Wang et al., 2013b, 2014c); it is the serial scanning of the aVLSI time window generator array that needs 25 cycles, as for any given time window generator, it has to be selected during the whole time slot. To maintain an architecture that is compatible with both the aVLSI and the digital time window generator array, we chose to use the time slot with 25 cycles for the work reported here.

Also at T9, the bistable weight will be updated to 1 if the weight is larger than a threshold, a pseudo random 4-bit number

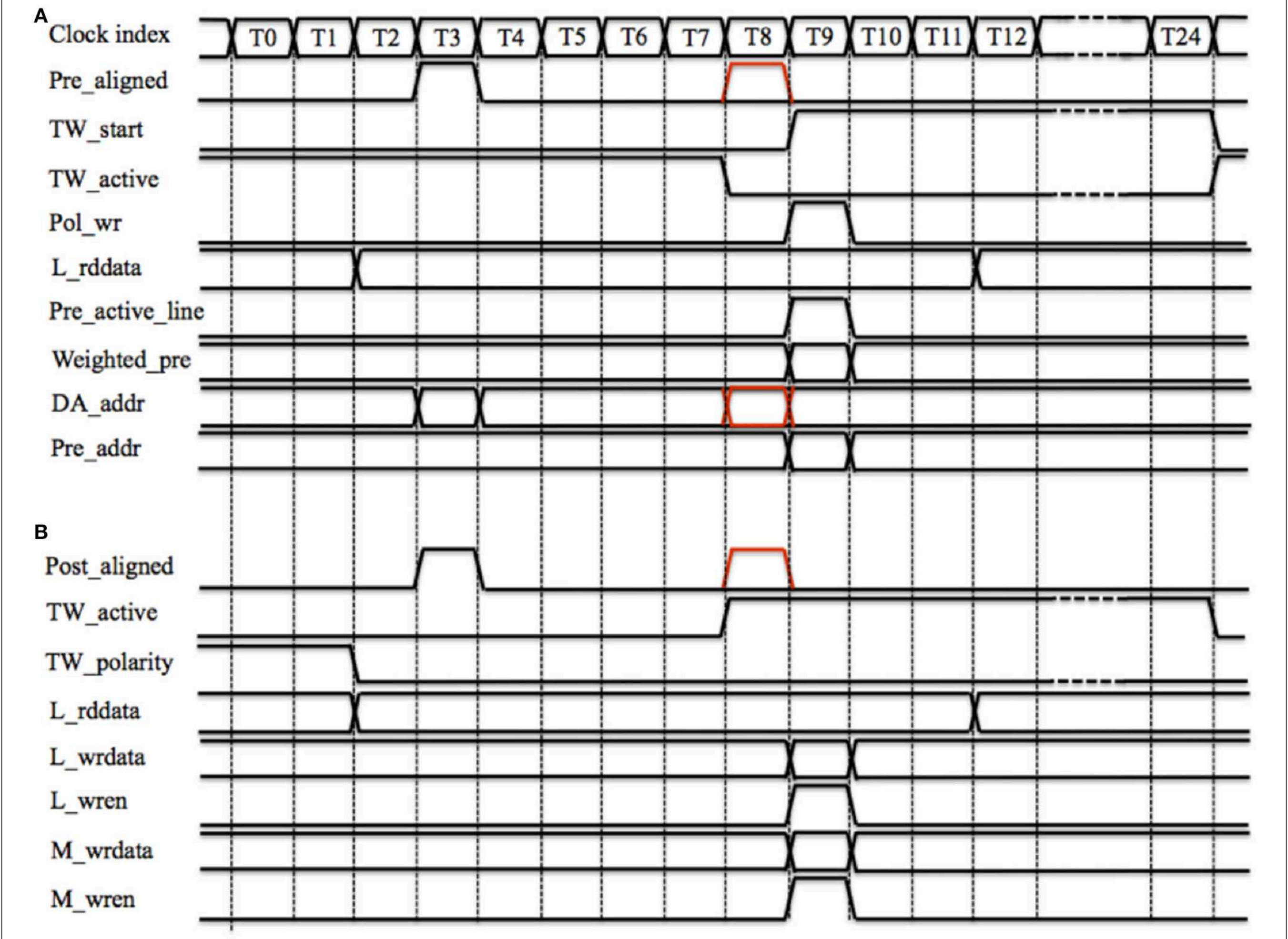


FIGURE 8 | TM STDP adaptor's timing diagram of one time slot. (A) Starting a time window. Pre_aligned is ready at T3 and TW_active will be ready at T8. As the time window is inactive, the delayed Pre_aligned (the red one) will start the time generator at T8 by sending a pulse (TW_start, starts at T9); **(B)** Increasing weight. The polarity of the time window (TW_polarity) and the local weight (L_rddata) are both read out at T0 and ready at T2. Post_aligned is ready at T3 and TW_active will be ready at T8. The delayed Post_aligned (the red one) will increase L_rddata.

between 4 and 11 that is updated every time slot. Otherwise, the bistable weight will be updated to 0. The updated bistable weight will be written into the Master RAM by asserting M_wren at T9.

TM STDDP Adaptor Array

The TM STDDP adaptor array operates in the same scheme (with the same pipeline stages) as the TM STDP adaptor array. From the controller's point of view, they are identical. This means that they are interchangeable, which was a deliberate design decision. **Figure 9** shows the structure of the TM STDDP adaptor array, which consists of a physical STDDP adaptor, a Local cache, an active RAM and a global counter. The Local cache, which is a dual port RAM with a size of 8 k × 4 bit, stores the 4-bit delay values of TM adaptors. The Master RAM stores the 4-bit delay values too. When a TM adaptor has been assigned to a DA adaptor, the delay of the latter will be read out from the Master RAM and then stored in the Local cache as the delay of that TM adaptor. When there is a modification of the delay the Master RAM is updated with the new delay.

Since the TM STDDP adaptor array pipeline is the same as the one presented for STDP earlier, the timing diagram is exactly the same as the ones presented in **Figure 8** (replacing "weight" with "delay") with the following additional changes:

1. Only the pre-synaptic spike can start the time window generator by sending it a spike with a duration proportional to the retrieved axonal delay.
2. The delayed pre-synaptic spike should be generated at the falling edge of the active line (from 0 × 1 to 0 × 0), which indicates the end of the axonal delay. Since this is a time multiplexing system, each TM adaptor will only know the value of the active line in the current time slot. To solve this problem, we introduced the active RAM, which is a dual port

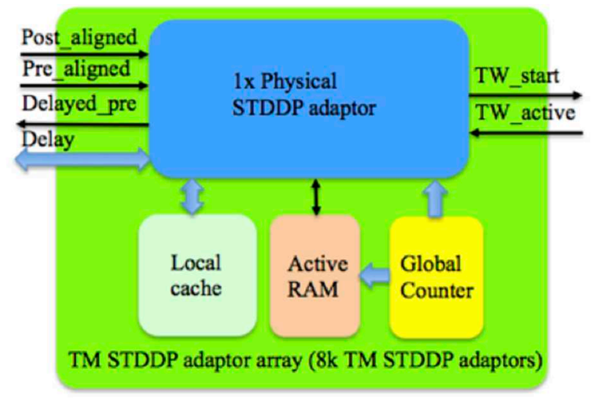


FIGURE 9 | Structure of the TM STDDP adaptor array. The global counter processes each TM STDDP adaptor sequentially. The Local cache stores the axonal delay values of the TM STDDP adaptor that are being processed. The learning rules will be performed with the aligned pre- and post- synaptic spikes from the controller and the active line from the time window generator.

RAM with a size of 8 k × 1 bit, to store the value of the active line in the current time slot. While the retrieved value from the active RAM represents the previous value. The delayed pre-synaptic spike will be generated if the active line is low and the active line retrieved from the active RAM is high. For this reason, the actual axonal delay will be from 1 to 16 ms while the value of the delay stored is from 0 × 0 to 0 × F. The signals of the polarity RAM (see **Figure 8**) are replaced by those of the active RAM. The weight of this spike will be a fixed but configurable value.

3. Only the post-synaptic spike can change the delay. No adaptation will be performed if the falling edge of the active line has been detected at T8 since this means the delay has been perfectly tuned and a delayed pre-synaptic spike will be generated at T9.

Utilization

The digital parts of the proposed array were developed using the standard ASIC design flow and therefore can be easily implemented with state-of-the-art manufacturing technologies. A bottom-up design flow was adopted in which we designed and verified each module separately. Once the module level verification was complete, all the modules were integrated together for chip-level verification. As a proof of concept, we implemented the proposed system on a Virtex6 XC6VLX240T FPGA, which is hosted on the Xilinx ML605 board. **Table 1** shows the utilization of hardware resources on the FPGA. Note that this is the utilization for the dynamically-assigned STDP/STDDP adaptor array (without the Master RAM), the digital time window generator array, and the interface circuit for the aVLSI time window generator. As **Table 1** shows, the proposed system uses only a small fraction (<1%) of the hardware resources. Limited by the size of the on-chip SRAM, for STDP and STDDP, we have implemented 1800 × 8 k = 14.4 M and 450 × 8 k = 3.6 M DA adaptors respectively. This is a proof of concept and in the future we will implement the Master RAM with off-chip memory, thus leveraging the design of the cache structure introduced.

TABLE 1 | Device utilization Xilinx Virtex6 XC6VLX240T.

Resource	STDP	STDDP	Total available
Occupied slices	558(1.4%)	545(1.4%)	37,680
Slice FF's	398(0.1%)	399(0.1%)	301,440
Slice LUTs	1430(0.9%)	1422(0.9%)	152,720
LUTs as logic	578(0.3%)	568(0.3%)	152,720
LUTs as RAM	827(1.4%)	827(1.4%)	58,400
36 k RAM	5(1.2%)	5(1.2%)	416

Results

For testing purposes, a PCB was developed as a daughter board to contain the aVLSI chip and was connected to the FPGA. The FPGA is controlled by a PC via a JTAG interface and the analog bias inputs of the aVLSI chip are controlled by external programmable bias voltages.

Performance of STDP

We have tested the performance of the dynamically-assigned STDP adaptor array by performing a balanced excitation experiment, based on the experiment run by Song et al. (2000). Song et al. (2000) have shown that competitive Hebbian learning (Hebb, 1949) can be performed through STDP. The competition (induced by STDP) between the synapses can establish a bimodal distribution of the synaptic weights: either toward zero (*weak*) or the maximum (*strong*) values.

Using Digital Time Window Generator Array

In this experiment, a single post-synaptic neuron is driven by 1024 TM synaptic adaptors, the TM addresses of which are from 0 × 0 to 0 × 3 FF. Their DA addresses are all the same: 0 × 0. That post-synaptic neuron has a single post-synaptic current generator that can generate both excitatory and inhibitory post-synaptic currents (EPSC and IPSC) modulated by the weights of the spikes arriving from different adaptors (Wang et al., 2014c). As the post-synaptic currents sum linearly in our model, only a PSC generator is needed in each neuron. Each adaptor was driven by an independent Poisson pre-synaptic spike train with the same average rate. We have tested the system with two firing rates: 10 and 20 Hz, whereas the firing rate of the post-synaptic neuron was 15 and 40 Hz respectively. The adaptors start with a uniform positive weight distribution. The size of the time window was fixed at 16 ms.

After 1.25 s of simulation, the distribution of synaptic weights converges to a steady-state condition with bimodal distribution of *strong* and *weak* weights (see **Figure 10**). Additionally, although our learning rule is considerably simplified when compared to that presented in Song et al. (2000), our system is capable of producing the same result: *for low input rates, more synaptic adaptors approach the upper limit, and for high input rates, more are pushed toward zero* (Song et al., 2000).

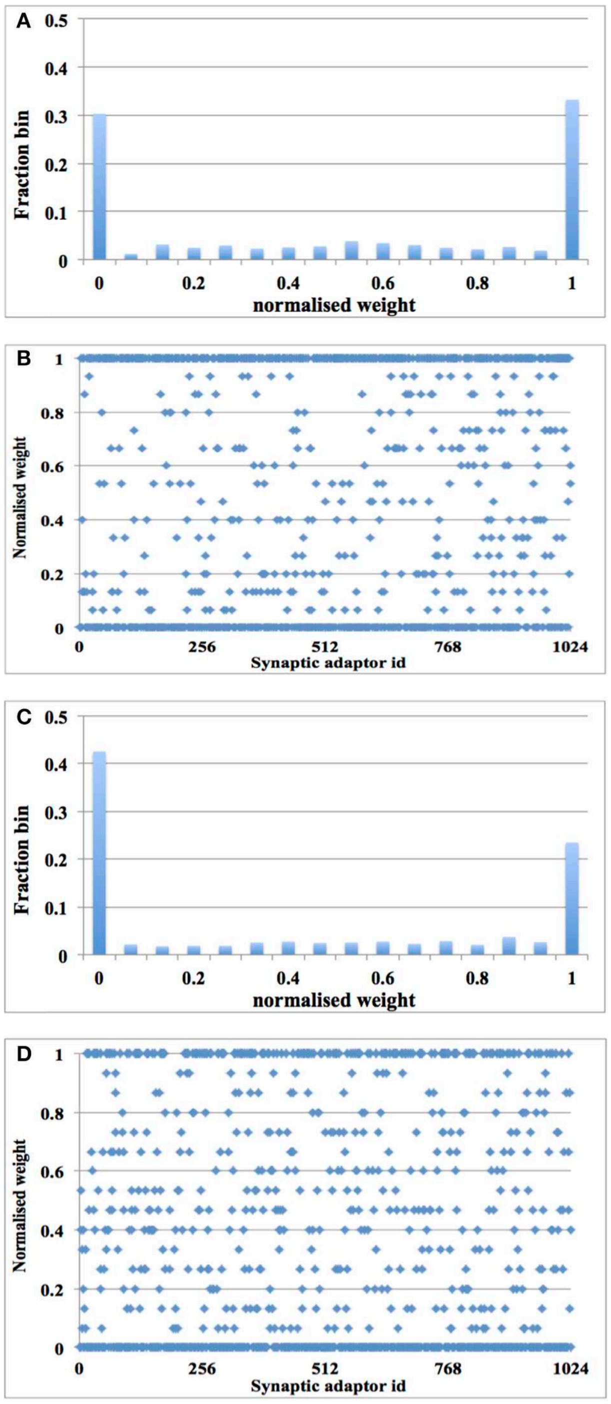


FIGURE 10 | Balanced excitation experiment with digital time window generator array. (A) Weight distribution after 1.25 s of STDP for an input rate of 10 Hz. The bimodal distribution of *strong* and *weak* weights is apparent; **(B)** Scatter plot of the final weight distribution; **(C,D)** Same as **(A,B)**, but for an input rate of 20 Hz. Now more weights are *weak* than *strong*.

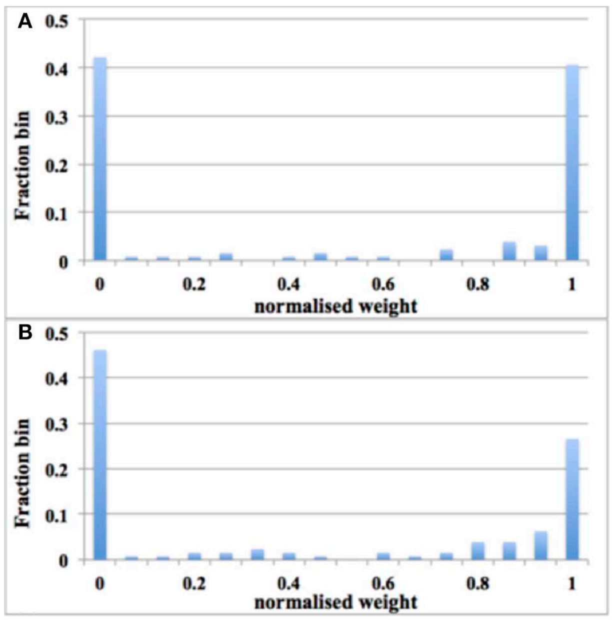


FIGURE 11 | Balanced excitation experiment with aVLSI time window generator array. (A) Weight distribution after 1.25 s of STDP for an input rate of 10 Hz. The bimodal distribution of *strong* and *weak* weights is apparent; **(B)** Same as **(A)**, but for an input rate of 20 Hz. Now more weights are *weak* than *strong*.

Using aVLSI Time Window Generator Array

We ran the experiment with 128 aVLSI time window generators (this is due to the fact that we have only 180 aVLSI time window generators and powers of two are preferable in digital design) with all the settings the same as with the digital time window generator. After 1.25 s of simulation, despite the adaptor using a fixed adaptation step (set to 1 here), the distribution of synaptic weights converges to a steady-state condition with a bimodal distribution of *strong* and *weak* weights (see **Figure 11**). It is also capable of producing the result: the higher the input rates, the more the synaptic weight will be pushed toward zero.

Validation of the Dynamic-assignment Scheme

The previous two experiments have shown that the balanced excitation experiment works for a system with 128 TM STDP adaptors. To validate the dynamic-assignment scheme, we conducted an experiment for 16 runs with an input rate of 20 Hz and 128 digital time window generators. For each run, these 128 TM STDP adaptors were assigned a DA address in the range from 0×0000 to 0×1 E00 with a step of 0×200. After each run, we read out the weights of these 128 adaptors (from the FPGA) and then started another run with the next DA address. In other words, we kept using the same 128 TM STDP adaptors for all the 16 runs by using the dynamic-assignment scheme. Note, this experiment is only a proof-of-concept and we can dynamically assign the TM adaptors for all those 8 k DA addresses (0×0 to 0×1 FFF) as long as the constraint of the active rate is not violated.

For each run, the distribution of synaptic weights converges to a steady-state condition with a bimodal distribution of *strong* and *weak* weights. **Figure 12** shows the average distribution of synaptic weights across all 16 runs. We first obtained the distribution of synaptic weights for each run and then averaged them. Since the input rate is 20 Hz, more synaptic weights

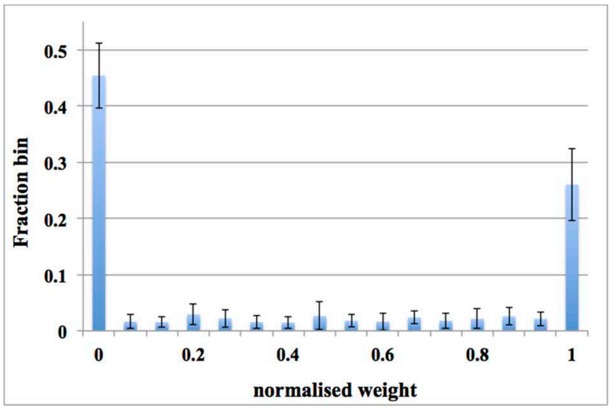


FIGURE 12 | Balanced excitation experiments using the dynamic-assignment scheme. One TM STDP adaptor array (with 128 TM STDP adaptors) was dynamically assigned for 16 DA STDP adaptor arrays. The averaged weight distribution after 1.25 s of STDP for an input rate of 20 Hz. Note these data are averaged across all 16 runs. The bimodal distribution of *strong* and *weak* weights is apparent and more weights are *weak* than *strong*. Error bars are standard deviations of 16 runs.

were pushed toward zero, which matches the results presented in **Figures 10C**, **11B**. For each run, the dynamic-assignment scheme has achieved a similar bimodal distribution of synaptic weights as the standard deviation of the results indicates. The dynamic-assignment scheme is therefore proved to be capable of performing what was designed to do: reusing hardware resources.

Performance of STDDP

We have tested the performance of the dynamically-assigned STDDP adaptor array by performing a polychronization experiment. The term polychronization is used to indicate that several neurons can fire asynchronously but after traveling along axons with specific delays, their spikes will arrive at a post-synaptic neuron simultaneously, causing it to fire in turn (Izhikevich, 2006). Neural networks based on this principle are referred to as “polychronous” neural networks and are capable of storing and recalling quite complicated spatio-temporal patterns. In Wang et al. (2014d), we have concluded that the most important requirement of a hardware implementation of a polychronous network is to provide a strong time-locked relationship. This is indeed the motivation for us to develop the STDDP learning rule, which will fine-tune the axonal delays to the desired delay values.

Using Digital Time Window Generator Array

In this experiment, we used 128 adaptors and a paired-pulse protocol: a single pair of pre- and post-synaptic spikes was sent to each of the adaptors periodically (every 32 time steps). During each period, each adaptor will receive one and only one pre-synaptic spike, the arrival time of which is randomized between time step 1 and 15. Additionally, during each period, each adaptor will receive one and only one post-synaptic spike, the arrival time is set to be time step 16. These spike pairs remain the same in each period. All the axonal delays are initialized to be zero. In each period, for each adaptor, a delay adaptation

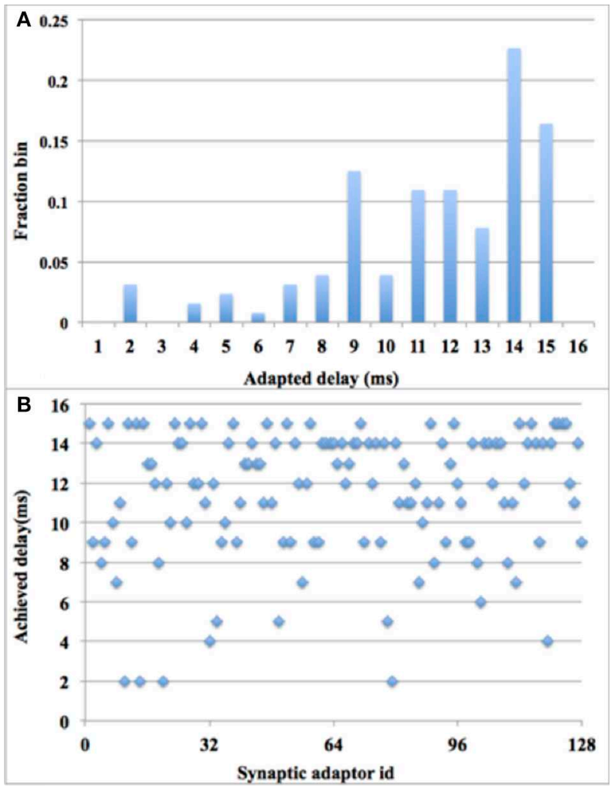


FIGURE 13 | Polychronization experiment with digital time window generator array. (A) Delay distribution after 15 times of STDDP; **(B)**. Scatter plot of the final delay distribution.

will be performed if the axonal delay has not been tuned to the desired delay. Hence, theoretically, after 15 times of STDDP, all the delayed pre-synaptic spikes from these 128 adaptors will fire simultaneously (each at its own time slot) at time step 16.

This theoretical behavior was confirmed via measurements on the FPGA. Since plotting 128 delayed pre-synaptic spike that fire at the same time is meaningless, we chose instead to show the delay distribution after 15 times of STDDP and the scatter plot of the final delay distribution (see **Figure 13**). It might be noticed that the final delays are not uniformly distributed, which indicated that more pre-synaptic spikes arrive at the early part of that period than the ones arrive at the later part. The system has performed the polychronization experiment successfully since all the axonal delays have been fine-tuned to the desired values, which are the time differences between pre- and post-synaptic spikes.

Using aVLSI Time Window Generator Array

The digital time window generator can generate any given desired size of the time window (from 1 to 15 ms, in a time-step of 1 ms). But due to process variation and device mismatch, it is impossible to tune all the aVLSI time window generators with such accuracy. To compare the performance with its digital counterpart, we tested the system with all the settings the same as with the digital time window generator conducting 10 test runs for statistical

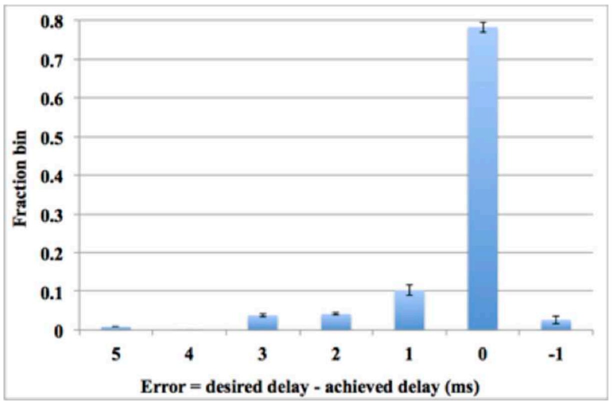


FIGURE 14 | Errors between the achieved delays and the desired values. Error bars are standard deviations of 10 runs.

purposes. **Figure 14** shows the errors between the achieved delays and the desired delays. Note these data are averaged across all 10 runs. As the results showed that 78% of the achieved delays match the desired delays perfectly (within one time step). This number will go up to ~91% when we counted in the achieved delays with an error of ± 1 ms, both of which will still contribute to the process of the polychronization (Wang et al., 2013b). Thus, only a small fraction of the achieved delays (less than 9%), will be unable to contribute to the network. The standard deviation of the results indicates that the aVLSI time window generator array has achieved a fair variability (the averaged stand deviation is 0.006).

Compared to our previous work that implemented the same STDDP learning rule with fully aVLSI circuits (Wang et al., 2014d), this mixed-signal solution achieved a much better performance in terms of accuracy and density. More importantly, this mixed-signal solution stores the axonal delays in the digital memory, which is non-volatile and much more compact. The work reported here was developed with the lessons that we have learnt from our previous work that suffered a lot from the intrinsic difficulties of the aVLSI circuits, e.g., coupling noises, leakage currents, process variations, and device mismatch.

Discussion

Since our goal aims for the maximum network size, our future work will focus on scaling up the network that we have presented here. For a system running at 266 MHz, we can achieve 256 k TM adaptors with one physical adaptor for a sub-millisecond time resolution. Given that we can implement 8 k DA STDP adaptors with a single TM adaptor, we can achieve 256 k × 8 k = 2G DA adaptors. With our bistable synaptic weight, which can be stored with a single bit, the total memory needed for this implementation is 2 Gb. It is clear that on-chip SRAM, which provides usually less than tens of megabits of storage, will not be able to meet this requirement. Among various external memory solutions, dynamic random access memory (DRAM) is the best candidate to provide the required storage because of its large storage capacity. High-end FPGA boards, such as Altera's DE5 board and Xilinx's VC709, usually contain two DDR3 SDRAM memories, each of which can currently support a maximum capacity of 64 Gb, and thus would allow us to implement 64G DA adaptors using only 64 physical adaptors. The corresponding TM adaptor arrays will need 64 × 8k × 4 bit = 2 Mb for the weight memory, which can easily be implemented using the on-chip SRAM. For the same system, the digital time window generator array would also need 2 Mb of storage.

In addition to the storage requirements, we also need to analyze the communications bandwidth requirement, which is generally the bottleneck for time-multiplexed implementations. The theoretically required bandwidth for 64 physical adaptors is 64 × 1 bit = 64 bits/clock cycle for both reading and writing. The DDR3 SDRAM is a single port device and the read/write operations cannot happen simultaneously. Thus, the required bandwidth of the SDRAM communication has to be doubled to 128 bits/clock cycle. Fortunately, the maximum theoretical bandwidth of one DDR3 SDRAM memory (when running at 1066 MHz) on an Altera DE5 board is 512 bits/clock cycle and even when considering that DDR3 memory typically only achieves 70% of that theoretical maximum bandwidth, there should be ample bandwidth to achieve the desired 128 bits/clock cycle. The reason for the reduced maximum bandwidth of the SDRAM is due to the need for flow control, which needs to take into consideration the bus turnaround time, memory refresh, finite burst length, and random access latency. All these will make the architecture of the system significantly more complex.

The cache structure was introduced to solve these difficulties. Firstly, it greatly reduces the bandwidth requirement to use external memory since the accessing (read/write) of the Master RAM will only be performed when needed and new values are not required to be available from memory every time slot. The reserved bandwidth can be used for other purposes, e.g., routing the spikes with look up tables. Secondly, this cache structure plus the fully-pipelined design style significantly ease the use of the off-chip memory. The pipeline of accessing the Master RAM can be simply reconfigured with different latency values to handle different flow control requirements.

The number of physical adaptors, i.e. the ones that can be activated simultaneously, will increase linearly with the number of available Slice LUTs, which are usually the bottleneck for high performance FPGA designs. But in our system, the design of the physical adaptor costs only a few LUTs and plenty of resources are left for additional physical adaptors or other systems components.

Based on the above calculations, we can conclude that it is practical to scale the proposed system up to a system with 64G DA adaptors on a commercial off-the-shelf high-end FPGA. The key to achieving this is to balance the number of physical adaptors (to achieve the best utilization of available hardware resources on the FPGA), the time-multiplexing rate (for a sub-millisecond time resolution), and the bandwidth and storage capacity of the memory.

Our aVLSI implementation is nowhere near as scalable as the digital implementation, since it can only be scaled up by implementing more physical copies of the aVLSI module. However, the introduction of the dynamic-assigning approach allows 8 k DA analog time window generators to be achieved with only a single physical time window generator. Above all, the motivation to develop the aVLSI implementation in the proposed system is for enhancement of the simulations.

Acknowledgments

This work has been supported by the Australian Research Council Grant DP140103001. The support by the Xilinx and Altera university program is gratefully acknowledged. This work was inspired by the Capo Caccia Cognitive Neuromorphic Engineering Workshop 2014 and Telluride Neuromorphic workshop 2014.

References

Amirikian, B., and Georgopoulos, A. P. (2003). Modular organization of directionally tuned cells in the motor cortex: is there a short-range order? *Proc. Natl. Acad. Sci. U.S.A.* 100, 12474–12479. doi: 10.1073/pnas.2037719100

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

Bliss, T. V., and Collingridge, G. L. (1993). A synaptic model of memory: long-term potentiation in the hippocampus. *Nature* 361, 31–39. doi: 10.1038/361031a0

Bofill-i-petit, A., and Murray, A. F. (2004). Synchrony detection and amplification by silicon neurons with STDP synapses. *IEEE Trans. Neural Netw.* 15, 1296–1304. doi: 10.1109/TNN.2004.832842

Brader, J. M., Senn, W., and Fusi, S. (2007a). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Brader, J. M., Senn, W., and Fusi, S. (2007b). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Brenner, S., and Sejnowski, T. J. (2011). Understanding the human brain. *Science* 334, 567. doi: 10.1126/science.1215674

Buskila, Y., Morley, J. W., Tapson, J., and van Schaik, A. (2013). The adaptation of spike backpropagation delays in cortical neurons. *Front. Cell. Neurosci.* 7:192. doi: 10.3389/fncel.2013.00192

Cassidy, A., and Andreou, A. G. (2008). "Dynamical digital silicon neurons," in *2008 IEEE Biomedical Circuits and Systems Conference*, (Baltimore, MD), 289–292.

Cassidy, A., Andreou, A. G., and Georgiou, J. (2011). "Design of a one million neuron single FPGA neuromorphic system for real-time multimodal scene analysis," in *2011 45th Annual Conference on Information Sciences and Systems*, (Baltimore, MD), 1–6.

Chicca, E., Badoni, D., Dante, V., D'Andreagiovanni, M., Salina, G., Carota, L., et al. (2003). A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. *IEEE Trans. Neural Netw.* 14, 1297–1307. doi: 10.1109/TNN.2003.816367

Galluppi, F., Davies, S., Furber, S., Stewart, T., and Eliasmith, C. (2012). "Real time on-chip implementation of dynamical systems with spiking neurons," in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (Brisbane, QLD: IEEE), 1–8.

Galluppi, F., Lagorce, X., Stromatias, E., Pfeiffer, M., Luis, A., Furber, S., et al. (2014). A framework for plasticity implementation on the SpiNNaker neural architecture. *Front. Neurosci.* 8:429. doi: 10.3389/fnins.2014.00429

Gao, C., and Hammerstrom, D. (2007). Cortical models onto CMOL and CMOS—architectures and performance/price. *IEEE Trans. Circ. Syst. I* 54, 2502–2515. doi: 10.1109/TCSI.2007.907830

Gerstner, W., Kempter, R., Van Hemmen, J. L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature* 383, 76–81. doi: 10.1038/383076a0

Giulioni, M., Camilleri, P., Dante, M. M., Braun, J., and Del Giudice, P. (2012). Robust working memory in an asynchronously spiking neural network realized with neuromorphic VLSI. *Front. Neurosci.* 5:149. doi: 10.3389/fnins.2012.00149

Goldberg, D., Cauwenberghs, G., and Andreou, A. (2001). Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons. *Neural Netw.* 14, 781–793. doi: 10.1016/S0893-6080(01)00057-0

Graupner, M., and Brunel, N. (2012). Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. *Proc. Natl. Acad. Sci* 109, 3991–3996. doi: 10.1073/pnas.1109359109

Häfliger, P. (2007). Adaptive WTA with an analog VLSI neuromorphic learning chip. *IEEE Trans. Neural Netw.* 18, 551–572. doi: 10.1109/TNN.2006.884676

Harkin, J., Morgan, F., Hall, S., Dudek, P., Dowrick, T., and McDaid, L. (2008). "Reconfigurable platforms and the challenges for large-scale implementations of spiking neural networks," in *2008 International Conference on Field Programmable Logic and Applications* (Heidelberg: IEEE), 483–486.

Harkin, J., Morgan, F., McDaid, L., Hall, S., McGinley, B., and Cawley, S. (2009). A reconfigurable and biologically inspired paradigm for computation using network-on-chip and spiking neural networks. *Int. J. Reconf. Comp.* 2009, 1–13. doi: 10.1155/2009/908740

Hebb, D. (1949). *The Organization of Behavior. Journal of Applied Behavior Analysis*. New York, NY: Wiley & Sons.

Holmgren, C., Harkany, T., Svennenfors, B., and Zilberter, Y. (2003). Pyramidal cell communication within local networks in layer 2/3 of rat neocortex. *J. Physiol. (Lond).* 551, 139–153. doi: 10.1113/jphysiol.2003.044784

Hubel, D. H., and Wiesel, T. N. (1974). Uniformity of monkey striate cortex: a parallel relationship between field size, scatter, and magnification factor. *J. Comp. Neurol.* 158, 295–305. doi: 10.1002/cne.901580305

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Izhikevich, E. M. (2006). Polychronization: computation with spikes. *Neural Comput.* 18, 245–282. doi: 10.1162/089976606775093882

Izhikevich, E. M., and Edelman, G. M. (2008). Large-scale model of mammalian thalamocortical systems. *Proc. Natl. Acad. Sci. U.S.A.* 105, 3593–3598. doi: 10.1073/pnas.0712231105

Johansson, C., and Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Netw.* 20, 48–61. doi: 10.1016/j.neunet.2006.05.029

Koickal, T. J., Hamilton, A., Tan, S. L., Covington, J. A., Gardner, J. W., and Pearce, T. C. (2007). Analog VLSI circuit implementation of an adaptive neuromorphic olfaction chip. *IEEE Trans. Circ. Syst. I* 54, 60–73. doi: 10.1109/TCSI.2006.888677

Lennie, P. (2003). The cost of cortical computation. *Curr. Biol.* 13, 493–497. doi: 10.1016/S

Liu, S., Kramer, J., Indiveri, G., Delbrück, T., and Douglas, R. (2002). *Analog VLSI: Circuits and Principles*. Cambridge, MA: MIT Press.

Magee, J. C. (1997). A synaptically controlled, associative signal for hebbian plasticity in hippocampal neurons. *Science* 275, 209–213. doi: 10.1126/science.275.5297.209

Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science (NY)* 275, 213–215. doi: 10.1126/science.275.5297.213

Minkovich, K., Srinivasa, N., Cruz-Albrecht, J. M., Cho, Y., and Nogin, A. (2012). Programming time-multiplexed reconfigurable hardware using a scalable neuromorphic compiler. *IEEE Trans. Neural Netw. Learn. Syst.* 23, 889–901. doi: 10.1109/TNNLS.2012.2191795

Mirhassani, M., Ahmadi, M., and Miller, W. C. (2007). A feed-forward time-multiplexed neural network with mixed-signal neuron–synapse arrays. *Microelectron. Eng.* 84, 300–307. doi: 10.1016/j.mee.2006.02.014

Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Circ. Syst.* 3, 32–42. doi: 10.1109/TBCAS.2008.2005781

Moore, S. W., Fox, P. J., Marsh, S. J. T., Markettosa, T., and Mujumdar, A. (2012). "Bluehive—a field-programable custom computing machine for extreme-scale real-time neural network simulation," in *20th IEEE International Symposium on Field-Programmable Custom Computing Machines*. (IEEE), 133–140. doi: 10.1109/FCCM.2012.32

Painkras, E., Plana, L. A., Garside, J., Temple, S., Galluppi, F., Patterson, C., et al. (2013). SpiNNaker: a 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid State Circ.* 48, 1943–1953. doi: 10.1109/JSSC.2013.2259038

Petersen, C. C. H., Malenka, R. C., Nicoll, R. A., and Hopfield, J. J. (1998). All-or-none potentiation at CA3-CA1 synapses. *Proc. Natl. Acad. Sci.* 95, 4732–4737. doi: 10.1073/pnas.95.8.4732

Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M., et al. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci.* 7:11. doi: 10.3389/fnins.2013.00011

Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., et al. (2012). Is a 4-bit synaptic weight resolution enough?—constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci.* 6:90. doi: 10.3389/fnins.2012.00090

Saighi, S., Levi, T., Belhadj, B., Malot, O., and Tomas, J. (2010). "Hardware system for biologically realistic, plastic, and real-time spiking neural network simulations," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, (Barcelona), 1–7.

Scannell, J. W., Blakemore, C., and Young, M. P. (1995). Analysis of connectivity in the cat cerebral cortex. *J. Neurosci.* 15, 1463–1483.

Schemmel, J., Fieres, J., and Meier, K. (2008). "Wafer-scale integration of analog neural networks," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, (Hong Kong), 431–438.

Sejnowski, T. J. (2012). Are we merely the sum of our neurons? *New Scientist* 213, 46. doi: 10.1016/S0262-4079(12)60317-0

Shi, Y., Veidenbaum, A. V., Nicolau, A., and Xu, X. (2015). Large-scale neural circuit mapping data analysis accelerated with the graphical processing unit (GPU). *J. Neurosci. Methods* 239, 1–10. doi: 10.1016/j.jneumeth.2014.09.022

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* 3, 919–926. doi: 10.1038/78829

Stanford, L. R. (1987). Conduction velocity variations minimize conduction time differences among retinal ganglion cell axons. *Science (NY)*, 238, 358–360.

Tsunoda, K., Yamane, Y., Nishizaki, M., and Tanifuji, M. (2001). Complex objects are represented in macaque inferotemporal cortex by the combination of feature columns. *Nat. Neurosci.* 4, 832–838. doi: 10.1038/90547

Vogelstein, R. J., Mallik, U., Vogelstein, J. T., and Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw.* 18, 253–265. doi: 10.1109/TNN.2006.883007

Wang, R., Cohen, G., Hamilton, T. J., Tapson, J., and Schaik, A. V. (2013a). "An improved aVLSI axon with programmable delay using spike timing dependent delay plasticity," in *2013 IEEE International Symposium of Circuits and Systems (ISCAS)* (Beijing: IEEE), 2–5.

Wang, R., Cohen, G., Stiefel, K. M., Hamilton, T. J., Tapson, J., and van Schaik, A. (2013b). An FPGA implementation of a polychronous spiking neural network with delay adaptation. *Front. Neurosci.* 7:14. doi: 10.3389/fnins.2013.00014

Wang, R., Hamilton, T. J., Tapson, J., and van Schaik, A. (2014a). "A compact reconfigurable mixed-signal implementation of synaptic plasticity in spiking neurons," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)* (Melbourne, VIC: IEEE), 862–865.

Wang, R., Hamilton, T. J., Tapson, J., and van Schaik, A. (2014b). "A generalised conductance-based silicon neuron for large-scale spiking neural networks," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE), 1564–1567.

Wang, R., Hamilton, T. J., Tapson, J., and van Schaik, A. (2014c). "An FPGA design framework for large-scale spiking neural networks," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)* (Melboune: IEEE), 457–460.

Wang, R., Jin, C., McEwan, A., and van Schaik, A. (2011a). A programmable axonal propagation delay circuit for time-delay spiking neural networks. *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, (Rio de Janeiro), 869–872.

Wang, R. M., Hamilton, T. J., Tapson, J. C., and van Schaik, A. (2014d). A mixed-signal implementation of a polychronous spiking neural network with delay adaptation. *Front. Neurosci.* 8:51. doi: 10.3389/fnins.2014.00051

Wang, R., Tapson, J., Hamilton, T. J., and van Schaik, A. (2011b). "An analogue VLSI implementation of polychronous spiking neural networks," in *2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing* (Adelaide, SA: IEEE), 97–102.

Wang, R., Tapson, J., Hamilton, T. J., and van Schaik, A. (2012). "An aVLSI programmable axonal delay circuit with spike timing dependent delay adaptation," in *2012 IEEE International Symposium on Circuits and Systems* (Seoul: IEEE), 2413–2416.

Weste, N., and Harris, D. (2005). *CMOS VLSI Design?: a Circuits and Systems Perspective. Energy Policy*, 3rd Edn., Vol. 24. Boston, MA: Addison-Wesley.

Wittie, L., Memelli, H. (2010). "Billion neuron memory models in slender Blue Genes," in *Program 208.30/MMM21, 2010 Neuroscience Meeting Planner* (San Diego, CA: Society for Neuroscience), 1.

Yu, T., and Cauwenberghs, G. (2010). "Log-domain time-multiplexed realization of dynamical conductance-based synapses," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, (Paris), 2558–2561.

Zaveri, M. S., and Hammerstrom, D. (2011). Performance/price estimates for cortex-scale hardware: a design space exploration. *Neural Netw.* 24, 291–304. doi: 10.1016/j.neunet.2010.12.003

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

frontiers in NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 20 January 2015
doi: 10.3389/fnins.2014.00429

A framework for plasticity implementation on the SpiNNaker neural architecture

Francesco Galluppi[1], Xavier Lagorce[1], Evangelos Stromatias[2], Michael Pfeiffer[3], Luis A. Plana[2], Steve B. Furber[2] and Ryad B. Benosman[1]*

[1] Equipe de Vision et Calcul Naturel, Vision Institute, Université Pierre et Marie Curie, Unité Mixte de Recherche S968 Inserm, l'Université Pierre et Marie Curie, Centre National de la Recherche Scientifique Unité Mixte de Recherche 7210, Centre Hospitalier National d'Ophtalmologie des quinze-vingts, Paris, France
[2] Advanced Processors Technology Group, School of Computer Science, University of Manchester, Manchester, UK
[3] Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland

Edited by:
Tara Julia Hamilton, University of Western Sydney, Australia

Reviewed by:
Siddharth Joshi, University of California, San Diego, USA
Jayram Moorkanikara Nageswaran, Brain Corporation, USA

***Correspondence:**
Francesco Galluppi, Vision Institute, Université Pierre et Marie Curie, Unité Mixte de Recherche S968 Inserm, l'Université Pierre et Marie Curie, Centre National de la Recherche Scientifique Unité Mixte de Recherche 7210, Centre Hospitalier National d'Ophtalmologie des quinze-vingts, 75012 Paris, France
e-mail: francesco.galluppi@inserm.fr

Many of the precise biological mechanisms of synaptic plasticity remain elusive, but simulations of neural networks have greatly enhanced our understanding of how specific global functions arise from the massively parallel computation of neurons and local Hebbian or spike-timing dependent plasticity rules. For simulating large portions of neural tissue, this has created an increasingly strong need for large scale simulations of plastic neural networks on special purpose hardware platforms, because synaptic transmissions and updates are badly matched to computing style supported by current architectures. Because of the great diversity of biological plasticity phenomena and the corresponding diversity of models, there is a great need for testing various hypotheses about plasticity before committing to one hardware implementation. Here we present a novel framework for investigating different plasticity approaches on the SpiNNaker distributed digital neural simulation platform. The key innovation of the proposed architecture is to exploit the reconfigurability of the ARM processors inside SpiNNaker, dedicating a subset of them exclusively to process synaptic plasticity updates, while the rest perform the usual neural and synaptic simulations. We demonstrate the flexibility of the proposed approach by showing the implementation of a variety of spike- and rate-based learning rules, including standard Spike-Timing dependent plasticity (STDP), voltage-dependent STDP, and the rate-based BCM rule. We analyze their performance and validate them by running classical learning experiments in real time on a 4-chip SpiNNaker board. The result is an efficient, modular, flexible and scalable framework, which provides a valuable tool for the fast and easy exploration of learning models of very different kinds on the parallel and reconfigurable SpiNNaker system.

Keywords: SpiNNaker, learning, plasticity, neuromorphic hardware, STDP, BCM

1. INTRODUCTION

Learning is crucial for the survival of biological organisms, because it allows the development of new skills, memories, and behaviors, in order to adapt to the information acquired from their local environment. Such high-level changes of behavior are the manifestation of an intricate interplay of synaptic plasticity processes, which lasts from early development throughout the adult life, and is taking place simultaneously and continuously in all parts of the nervous system. Although neuroscience has developed an increasingly better insight into the local plasticity mechanisms at specific types of synapses, we still have a poor understanding of the global effects of plasticity that lead to the emergence of our astonishing cognitive capabilities. Clearly, this is one of the great unsolved questions, not only for neuroscience, but with great implications for fields like philosophy, psychology, medicine, and also for engineering disciplines concerned with the development of artificial intelligent systems that can learn from their environment.

Much of our understanding of the functional effects of local plasticity comes from theoretical and simulation studies of simplified learning rules in neural network models. Most influential is the hypothesis of Hebb (1949), which says that synaptic connections strengthen when two connected neurons have correlated firing activity. This has inspired many classical models for associative memory (Hopfield, 1982), feature extraction (Oja, 1982), or the development of receptive field properties (Bienenstock et al., 1982). Later, the discovery of Spike-timing Dependent Plasticity (STDP) (Markram et al., 1997; Bi and Poo, 1998) has led to a number of models that have exploited the precise timing properties of spiking neurons for receptive field development (Song and Abbott, 2001; Clopath et al., 2010), temporal coding (Gerstner et al., 1996; Guyonneau et al., 2005), rate normalization (Song et al., 2000; Kempter et al., 2001), or reward-modulated learning (Izhikevich, 2007; Legenstein et al., 2008; Friedrich et al., 2011; Potjans et al., 2011). It has also been realized that there is not one standard model for STDP, but that there is a huge diversity of learning rules in nature, depending on species, receptor, and neuron types (Abbott and Nelson, 2000; Kullmann et al., 2012), the presence or absence of neuromodulators (Pawlak et al., 2010; Cassenaer and Laurent, 2012), but also on other factors like post-synaptic membrane potential, position on the dendritic arbor, or synaptic weight (Sjöström et al., 2001).

The discovery that basic effects can be achieved with local learning rules has had a big influence on the development of larger scale learning models that have mapped methods from machine intelligence onto spiking neural networks. Examples include supervised learning methods for classification of visual (e.g., Brader et al., 2007; Beyeler et al., 2013), or auditory stimuli (Sheik et al., 2012), and unsupervised learning methods like Expectation Maximization (Nessler et al., 2013; Kappel et al., 2014), Independent Component Analysis (Savin et al., 2010), or Contrastive Divergence (Neftci et al., 2014). This has opened up the possibility of using spiking neural networks efficiently for machine learning tasks, using learning algorithms that are more biologically plausible than backpropagation-type algorithms typically used for training artificial neural networks.

The increased interest in spiking neural networks for basic research and engineering applications has created a strong interest for larger, yet computationally efficient simulation platforms for trying out new models and algorithms. Being able to easily and efficiently explore the behavior of different learning models is a very desirable characteristic of a such platform. The major problem for computation with spikes is that it is a resource-intensive task, due to the large number of neurons and synapses involved. Synaptic activity, and specifically synaptic plasticity, which might be triggered by every spike event, is dominating the computing costs in neural simulations (Morrison et al., 2005; Brette et al., 2007), partly because the communication and processing of large numbers of small messages (i.e., spikes), is a bad match for current von Neumann architectures. Different strategies to improve the scale and run-time efficiency of neural simulations either rely on supercomputer simulations (Plesser et al., 2007; Wong et al., 2013), parallel general-purpose devices such as GPUs (Fidjeland and Shanahan, 2010) and FPGAs (Neil and Liu, 2014), or special purpose *neuromorphic* hardware (Indiveri et al., 2011). Each solution involves a trade-off between efficiency, reconfigurability, scalability and power consumption.

In this context we present a framework for studying arbitrary plasticity models on a parallel, configurable hardware architecture such as SpiNNaker. The SpiNNaker system (Furber et al., 2006, 2014) has been designed as a massively parallel, highly reconfigurable digital platform consisting of multiple ARM cores, which optimally fits the communication requirements for exploring diverse synaptic plasticity models in large-scale neural simulations. Previous implementations of plasticity on SpiNNaker have been limited in their ability to model arbitrary spike- and rate-based learning rules. Here, we present a new approach for implementing arbitrary plasticity models on SpiNNaker, using a dedicated plasticity core that is separated from other cores that process other neural and synaptic events. Specifically we demonstrate the implementation of three synaptic plasticity rules with very different requirements on the trigger events, and on the need to store or access additional variables for computing the magnitude of updates. We show that the same architecture can implement the rate-based BCM rule (Bienenstock et al., 1982), an implementation of standard STDP based on a model by Morrison et al. (2008), and a voltage-dependent STDP rule suggested by Brader et al. (2007). We compare the efficiency and correctness of the STDP rule to previous implementations on SpiNNaker, and provide the first implementation of BCM and the learning rule of Brader et al. (2007) on this platform. All the experimental results presented in this paper come from implementations of learning rules on a 4-chip SpiNNaker board.

The ability to implement different rules with very different requirements, that are either based purely on spike-timing, on the correlation of firing rates, or on additional voltage signals indicates that the framework can be used as a generic way of implementing plasticity in neural simulations. This new architecture therefore provides an efficient way for exploring new network models that make use of synaptic plasticity, including novel rules and combinations of different plasticity rules, and paves the way toward large-scale real-time learning systems.

This article is organized as follows: the next Section introduces different approaches to model learning, from a theoretical and an implementation point of view. Section 3 describes the SpiNNaker system, the previous solutions for plasticity on SpiNNaker and our novel approach presented in this work. The flexibility of the framework introduced is demonstrated by the implementation of three different rules, presented in Section 4, 5, and 6: Spike-Timing Dependent Plasticity (Gerstner et al., 1996), the rate-based BCM rule (Bienenstock et al., 1982) and the voltage-dependent variation of the STDP rule (Brader et al., 2007). We validate the implementation by replicating classical plasticity experiments, and discuss the performances of each rule in Section 7. The paper is concluded in Section 8, which also provides an outlook toward future applications.

2. LEARNING IN SPIKING PLATFORMS

The use of parallelization to mitigate the computational costs and difficulties of modeling large plastic networks has been exploited using different tools and strategies. Using many processors in a supercomputer is an important exploratory solution, which can be used to rapidly implement and test learning rules. However, setting up a Message Passing Interface (MPI) mediating the spike communication is a challenging process on a distributed von-Neumann architecture, because the network infrastructure is optimized for large-frame transfers (Plesser et al., 2007; Wong et al., 2013) as opposed to small spike packets.

Dedicated neuromorphic (Mead, 1989) systems are natural candidates for emulating parallel neural computation. On these systems, circuits modeling neurons and synapses can be replicated using Very Large Scale Integration technology (VLSI, Indiveri et al., 2011). Synapses usually take up the majority of the resources, in terms of computation and chip area. It is also particularly challenging to design plastic hardware synapses. In the FACETS wafer-scale hardware (Schemmel et al., 2007), for example, the area of plastic synapses is minimized by separating the accumulator circuit for the spike-timing dependency and a global weight-update controller, which drives the update of multiple synapses (Pfeil et al., 2012). Having a separate plasticity engine makes the update slower, but adds flexibility to the plasticity algorithms that can be implemented. The trade-off in this case relates to the controller frequency update, which evolves slower than the neural dynamics, and the precision of the synapses, limited to 4-bits. Despite these limitations the system is capable of modeling a variety of plasticity models, characterized by different weight dependencies. Also, the synaptic resolution is shown to be not-critical in the simulation of a series of network benchmarks. Vogelstein et al. (2002) have introduced a general system where synapses are stored in digital memory with a processor implementing the synaptic update mechanism, while a separate set of ASICs implement the neural integration process. While they demonstrate STDP, more general functions can be implemented using the same scheme.

Brader et al. (2007) proposed a learning rule that captures biological properties such as memory preservation and encoding. Furthermore, it is optimized for efficient implementation in a neuromorphic system. The rule is dependent on the post-synaptic neuron membrane potential and recent spiking activity at the time of a pre-spike arrival. Every synapse has internal dynamics, which drives the weight toward a bistable state. Its advantage for VLSI implementations (Indiveri et al., 2006; Giulioni et al., 2008; Mostafa et al., 2014) lies in its ability to smooth device mismatch by applying a threshold to the internal state variable, in order to set the synapses to one of two possible states. The bistable representation of memory has the additional advantage of being power efficient. The fact that the rule can be computed when a pre-synaptic spike is received reduces the chip area required by a synapse, and consequently increases the number of synapses that can be modeled. This assumes that the synapses are located on the post-synaptic neuron, and have access to the neural and synaptic state variables when a spike is received. This is the case in the VLSI devices mentioned above and also in SpiNNaker (Jin et al., 2010a). A review of different neuromorphic approaches and challenges in designing plastic synapses can be found in Rahimi Azghadi et al. (2014), which discusses power consumption, area requirements, storing techniques, process variation and device mismatch.

Recently, Resistive Random Access Memories, commonly referred as memristors, have raised interest in the neuromorphic community. They are small, power-efficient devices that can be used to store weights and thereby increase the amount of neurons and synapses that can be integrated in a chip. Weight change can be induced by controlling the voltage at the terminals of a memristor, inducing a change in its state and thus modeling a learning rule such as STDP (Zamarreño Ramos et al., 2011) or triplet-based STDP (Mayr et al., 2012). In Indiveri et al. (2013) memristors are used directly to model synaptic dynamics, using them both for computation and memory storage.

There are also difficulties when implementing synaptic plasticity in general purpose hardware. Regarding GPUs Fidjeland and Shanahan (2010), for example, propose a simplified nearest-neighbor pairing scheme with a time-limited STDP window. They continuously accumulate STDP statistics that are then used to update synapses at fixed intervals. In such implementation, increasingly shorter intervals impact performance, lowering the overall spike throughput of the platform. Weight change accumulation is commonly used in other GPU approaches, e.g., in Nageswaran et al. (2007), where the synaptic kernel update is applied every second, and in software simulations (Izhikevich, 2006).

The diversity of approaches for studying synaptic plasticity in hardware, indicates a need for general purpose, massively parallel, and reconfigurable computing platforms. Only this will allow fast prototyping of plasticity rules, and their exploration in large scale models, which can in a second stage directly lead to dedicated hardware implementations.

3. A NOVEL FRAMEWORK FOR PLASTICITY IMPLEMENTATION ON SPINNAKER

SpiNNaker (Furber and Temple, 2008; Furber et al., 2014) is a digital multi-core, multi-chip architecture designed for the simulation of large neural networks in real time. Each SpiNNaker chip is equipped with a 1Gbit SDRAM and 18 programmable ARM968 cores embedded in a congurable packet-switched asynchronous fabric (Plana et al., 2007).

The SpiNNaker network infrastructure is designed with spiking communication in mind: every chip contains an on-chip Multicast (MC) router capable of handling very efficiently one-to-many communication of spikes (MC packets). The router links every chip to its six neighbours. Each core has a small local tightly-coupled memory (32 kByte instruction and 64 kByte data, ITCM and DTCM respectively). The massive synaptic data required for neural simulations is stored in the shared, off-die SDRAM 128 MByte chip that can be accessed by the cores through DMA requests, for an aggregate read/write bandwidth of 900 MBytes/s (Painkras et al., 2013). The system is designed to scale up to 60,000 chips for a total of over one million ARM cores. The goal of the system is to simulate 1% of the human brain in real time.

A high level view of the main chip components is presented in **Figure 1**. When simulating neural networks, spikes are delivered and processed by the ARM cores, which update the states of the neurons and synapses. A C-based API is used to program neural kernels (Sharp et al., 2011). The API offers an accessible interface to the hardware substrate and to real-time event scheduling facilities, and can be used to write applications that are executed in parallel on the machine. The API promotes an event-driven programming model: the neural kernels are loaded into the ARM cores and are used to configure *callbacks* that respond to *events*. A timer event allows the periodic execution of functions, such as neuron state update. A packet event signals the arrival of an MC packet (spike) and can be used to initiate a request to transfer synaptic data from SDRAM. Finally, a memory event indicates that the requested data is available for processing. The neural kernels are parameterizable and can support different classes of neural models and connectivity patterns. Model specification, system mapping and run-time control is obtained through the PArtition and Configuration MANager (PACMAN, Galluppi et al., 2012), which offers interfaces with two languages extensively used in the neural modeling community: PyNN (Davison et al., 2008), a simulator-independent specification language, and Nengo (Stewart et al., 2009), the simulation tool implementing the principles of the Neural Engineering Framework.

Figures 2A,B shows the current implementation of a neural kernel, highlighting the processes involved: every millisecond, a *timer event* triggers the evaluation of the neural dynamics. A spike is then emitted if a configurable threshold of the membrane potential has been reached. Spikes travel as MC packets through routers on the interconnection fabric and are delivered to the destination cores, triggering a *packet event*. Whenever a packet is received, a memory look-up is initiated to retrieve the relevant synaptic information (such as weight, delay, destination neurons on the core, and type of synapse) from SDRAM, where the connectivity matrix, indexed by pre-synaptic neuron, is stored. When the requested data arrives this creates a *memory event*, and the spike is processed by every post-synaptic core. Due to the limited memory available in the ARM cores, the synaptic weights are only locally available to the core right after a memory transfer from DMA has occurred as a consequence of the arrival of a spike. Therefore, the time available for the weight update process is very short; moreover, since delays are reintroduced at the post-synaptic end, the update process relies on information which might concern the future state of the neuron. This has limited the flexibility of previous approaches for implementing plasticity on SpiNNaker.

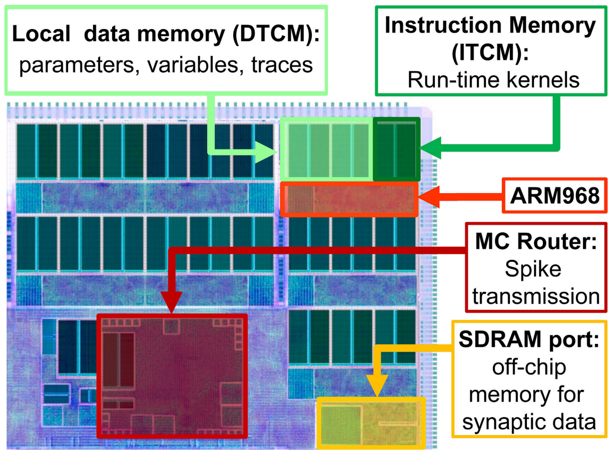

Local data memory (DTCM):
parameters, variables, traces
Instruction Memory (ITCM):
Run-time kernels
ARM968
MC Router:
Spike transmission
SDRAM port:
off-chip memory for synaptic data

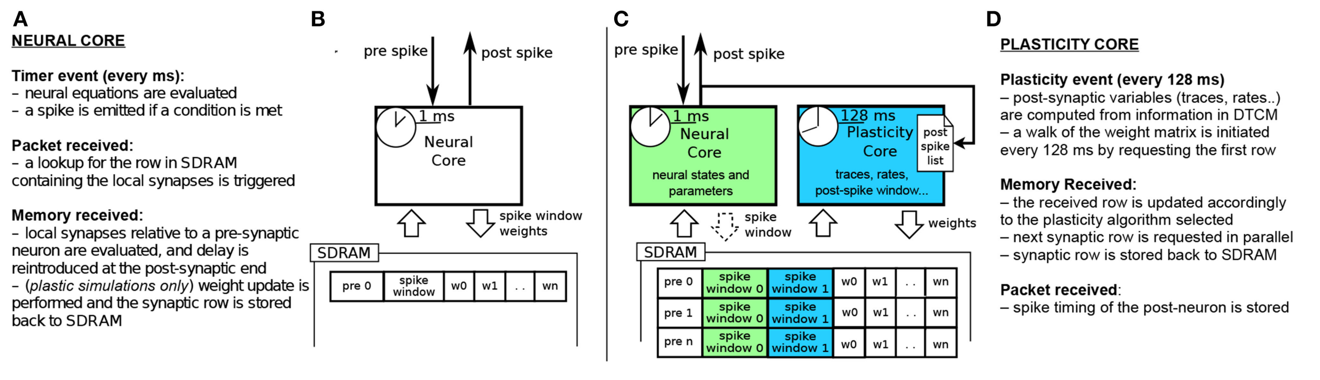
FIGURE 1 | High-level view of the SpiNNaker chip, showing: the ARM cores with their Instruction and Data Tightly coupled memory (DTCM and ITCM, 32 and 64 Kbyte respectively) to run applications and locally store data; the Multicast (MC) router responsible of spike transmission; the port to the 128 Mbyte SDRAM off-chip memory, containing the synaptic data.

3.1. THE DEFERRED EVENT DRIVEN MODEL

The STDP algorithm requires computation whenever a pre spike is received or a post spike is emitted. This causes two relevant issues for the cores running neural simulation on SpiNNaker:

1. Weights are only available in local memory upon the reception of a MC packet signaling that a spike has occurred in one of the pre-synaptic neurons. At the time of a post-synaptic spike such information is stored in SDRAM, which is indexed by pre-synaptic neuron and therefore is not easily accessible for a fast update.
2. A spike packet is delivered to the post-synaptic core as soon as it is emitted, and biological delays (stored in SDRAM as well) are re-introduced by the core modeling the post-synaptic neuron *after* the relevant information has been retrieved from memory; the delay itself is stored into memory, and can be different for different post-synaptic neurons on the same core (Jin et al., 2010a). The weight value is stored in a circular buffer which rotates with the timer event interval, and lumps all the synaptic contributions for one millisecond in a way similar to that described in Morrison et al. (2005). The consequence of delaying the input into the future is that when a synapse is processed, the state of the post-synaptic neuron (e.g., its membrane potential or the presence of a post-synaptic spike) is not available.

The *Deferred Event Driven Model* (DED) for computing plasticity was introduced in Jin et al. (2009) to circumvent these problems. DED enables computation of STDP at the time when a pre spike is received by deferring the weight update process into the future, until enough information is gathered. Post spikes are collected in a spike window, stored in the local core memory, while pre spikes are stored in SDRAM, along with the rest of the synaptic information. Upon the retrieval of the weights related to a pre spike,



A
NEURAL CORE

Timer event (every ms):
– neural equations are evaluated
– a spike is emitted if a condition is met

Packet received:
– a lookup for the row in SDRAM containing the local synapses is triggered

Memory received:
– local synapses relative to a pre-synaptic neuron are evaluated, and delay is reintroduced at the post-synaptic end
– (*plastic simulations only*) weight update is performed and the synaptic row is stored back to SDRAM

D
PLASTICITY CORE

Plasticity event (every 128 ms)
– post-synaptic variables (traces, rates..) are computed from information in DTCM
– a walk of the weight matrix is initiated every 128 ms by requesting the first row

Memory Received:
– the received row is updated accordingly to the plasticity algorithm selected
– next synaptic row is requested in parallel
– synaptic row is stored back to SDRAM

Packet received:
– spike timing of the post-neuron is stored

FIGURE 2 | **(A,B)** current STDP implementation on SpiNNaker, following the Deferred Event Driven model **(C,D)** the proposed novel implementation framework for plasticity implementation.

these two time windows are compared and weight update is performed. Plasticity is therefore always computed on the *next* pre spike arrival, and only if enough time has passed, to guarantee that all the necessary information is available. This poses restrictions on the pre-to-post firing rates: if a pre-synaptic neuron fires with a low rate, the spike information of the post-synaptic neuron might have already expired. Thus, the algorithm loses a pre-post spike pair, even if they were close in time, if the next pre spike arrives after the expiration of the post-spike window. Furthermore, because the algorithm needs to check every spike pair, its efficiency depends on the length of the history and on the number of the pre-post pairs. Such limitations are discussed in Jin et al. (2010b), where the trade-offs between spike-history, efficiency and correctness are analyzed.

Davies et al. (2012) try to address the problem from a different angle, using the *Time To Spike* (TTS) strategy: STDP is computed only upon the reception of a pre spike, using the current membrane potential as a predictor of future spiking activity. By doing so, weight updates can be performed while synaptic information is in local memory, addressing the first of the two problems mentioned above. However, as mentioned earlier, spikes are delivered to the post-synaptic neurons as soon as they are emitted, and the biological delays are reintroduced at the post-synaptic end. This creates errors when using delays, as reported in the original work presenting the TTS approach: the membrane potential used as a predictor of the post neuron firing is the one corresponding to the time of spike *emission* by the pre neuron, rather than that of spike *reception* by the post neuron (after the propagation delay). Such problem makes the TTS algorithm usable and efficient when delays are constant and short, but cannot deal correctly with longer delays. This also creates problems for detecting temporal patterns where delays play an important role (Izhikevich, 2006), such as in the experiments in Section 6.2.

3.2. THE DEDICATED PLASTICITY CORE APPROACH

The previous implementations of plasticity are not limited by the SpiNNaker hardware, but rather by their software implementation. Therefore, we present an alternative approach: instead of having a single core evaluating neural dynamics and plasticity, we divide the job into two parallel processes. One core performs the neural updates and spike integration, while the second core deals with plasticity (see **Figures 2C,D**). Plasticity operates as a slower process in the background. It processes the whole synaptic block in SDRAM and the information about spike timing, and modifies the weights according to the chosen plasticity mechanism. The proposed approach takes inspiration from previous work where plasticity effects are accumulated and evaluated periodically (Izhikevich, 2006; Nageswaran et al., 2007; Fidjeland and Shanahan, 2010; Pfeil et al., 2012). Plasticity is thus updated less frequently than neural dynamics, which is radically different from the previously described DED model on SpiNNaker.

In our novel approach, the PACMAN mapping tool automatically instantiates a *twin* plasticity core alongside each neural core whenever it detects a neural population with incoming plastic connections. Neural and plasticity cores have access to the same portion of SDRAM through replication, in their local memories, of the look-up tables used to index it. The neural core performs the usual operation that a non-plastic core would perform, thus eliminating all the overheads required by the DED model. The neural core is also in charge of trivially updating a bitmap pre-spike window whenever a pre spike is received, as shown by the dashed arrow in **Figure 2C**. The plasticity core is concerned solely with the weight update process, which can be performed by walking the local SDRAM weight matrix and computing plasticity at a slower pace. When a neuron in the neural core emits a spike, the corresponding packet is delivered to the plasticity core, and to the post-synaptic neurons as under normal conditions. Because the plasticity and neural core always reside on the same chip, this process does not add overhead to the routing process. This allows to keep track of the post-synaptic spiking history. Here we decided to update the weights every 128 ms and store the spike times with a resolution of 2 ms, as a compromise between performance, platform-specific limitations and precision. Pre-synaptic spikes are stored at the beginning of each synaptic row as spike-history bitmaps. The plasticity process needs to know all the spikes which happened in its considered 128 ms window. This data has been stored by the neural core in one of the spike windows (0 or 1 in the Figure) during the previous 128 ms before the update. For the plasticity core to be able to read this buffer while the neural core is storing the next 128 ms of spikes, we use a double buffer technique: when the plasticity core is reading spike window 0, the neural core is storing the spikes in spike window 1 and viceversa. This has been emphasized in the **Figure 2C** by using different color codes for the two different processes. The double buffers contain data for different time slots and therefore do not need to be accessed concurrently by the neural and plasticity core, so there is no need for mutual exclusion or locks. Memory contention is eliminated by the fact that the neural core operates in the *current* 128 ms window, while the plasticity core works in the *previous* 128 ms time window. The same technique used for the spike windows could be used on the whole synaptic matrix to ensure coherency of the whole matrix during the entire simulation. Because this method only switches the pointer used to lookup the data between consecutive plasticity periods, this would not change the approach or performances. Whenever a portion of memory is ready for computation, the request for the next row of the synaptic matrix is issued and weight updates of the current synaptic row are performed, thus masking memory access costs through parallelization. This separation of neural and plasticity operations gives rise to an environment where weight update rules can be easily programmed separately. This leverages the reprogrammability of the general processors used by SpiNNaker and the generality of the event-driven API presented in Sharp et al. (2011). The general infrastructure for the framework is presented in Appendix A. While it is worth noting that the difference between neural and the plasticity processes is only in the software running on the ARM cores, they can be thought of as hardware threads. The SpiNNaker software infrastructure does not support threads. If software threads were available, besides the costs related to thread switching, the neural and synaptic update threads would need to split between them the limited local memory (DTCM) and the processor cycles. In SpiNNaker, clock cycles are also limited in order to meet real-time targets. The proposed solution, on the other hand, uses hardware threads (cores), one for neural update and one for synaptic update, with each thread owning all of its local resources. This results in a more efficient use of the available resources. In fact, depending on the relative complexity of the neural and synaptic update processes, the ratio of hardware threads can be adjusted, using N neural update for every M synaptic update threads (cores). The plasticity core has access to the pre- and post-synaptic spike activity history of the previous 128 ms time window; the first is stored in SDRAM and the second one in DTCM. Such information can be used to compute rates, traces, timing differences or other required variables for different learning rules, as shown by the three rules implemented in this paper.

4. STDP

Derived from biological observations that synaptic plasticity depends on the relative timing of pre- and post-synaptic spikes (Markram et al., 1997; Bi and Poo, 1998), Spike-Timing Dependent Plasticity (STDP) (Gerstner et al., 1996; Song et al., 2000) has become a popular model for learning in spiking neural networks. In its standard form, STDP weight-updates are expressed by the double-exponential form

$$F(\Delta t) = A_{+} e^{\frac{\Delta t}{\tau_{+}}} \qquad \Delta t < 0 \tag{1}$$

$$F(\Delta t) = -A_{-} e^{\frac{-\Delta t}{\tau_{-}}} \qquad \Delta t \geq 0 \tag{2}$$

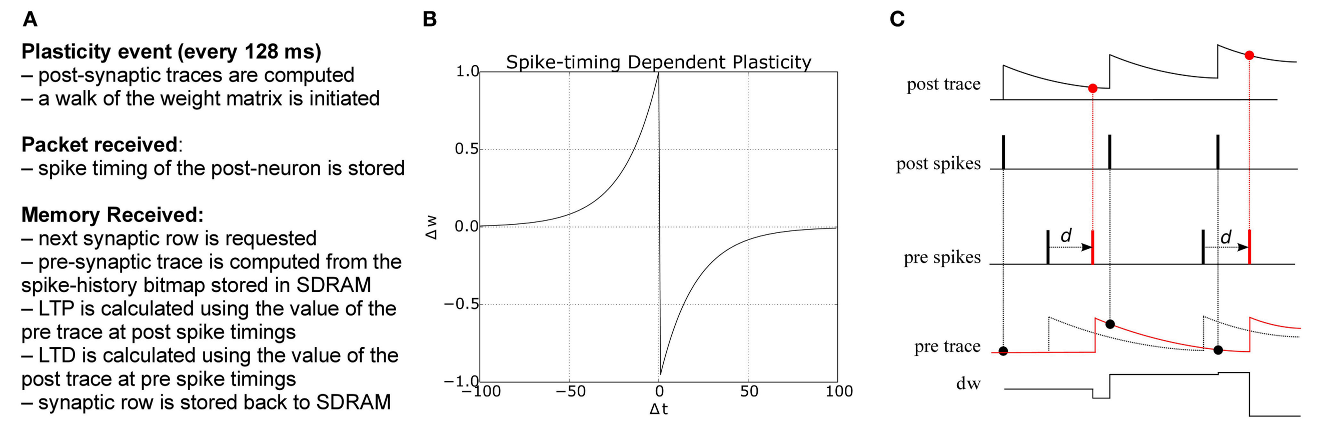
where $\Delta t = t_{\text{pre}} - t_{\text{post}}$ is the time difference between a pair of pre- and post-synaptic spikes, A_{+} and A_{-} are scaling factors for potentiation and depression, and τ_{+} and τ_{-} are the time constants of the plasticity curves. The weight update rule is illustrated in **Figure 3**. There are different strategies for computing the total amount of weight change after seeing multiple pre- and post-synaptic spikes (Morrison et al., 2008), e.g., by considering only nearest neighbor spike pairs, or summing the weight changes $F(\Delta t)$ for all pairs. Here we adopt a form of STDP proposed by Morrison et al. (2008) to compute the weight change using local variables in the form of pre- and post-synaptic traces. Each trace x_i has the form

$$\frac{dx_i}{t} = -\frac{x_i}{\tau} + A \sum_{t_i^f} \delta(t - t_i^f), \tag{3}$$

where x_i is the value of the trace for neuron i, A is the amplitude by which the trace increases with each new spike at time t_i^f, and τ is the exponential decay time constant. The concept is illustrated in **Figure 3**: potentiation occurs at post-synaptic spikes, using the value of the pre-synaptic trace as the weight increase; conversely, depression happens at pre-synaptic spike times, and reduces the weight by the value of the post-synaptic trace.

4.1. METHODS: IMPLEMENTATION OF STDP ON THE PLASTICITY CORE

The plasticity core is in charge of computing all traces, using the spike timing information collected during the simulation. Weight changes are then computed by walking through all the synaptic block. The pre-trace is computed every time a portion of memory is received through a DMA process using the information in the spike window, while the post trace is computed at the beginning of each plastic phase starting from the spike history bitmap collected during the packet received callback. Traces can have longer time scale than the plasticity window, as the exponential filtering is updated at the beginning of each phase, and the previous value of the exponential filter carries over from one plasticity window to the next. Delay needs to be reintroduced at the post-synaptic end, and can be used to compute the amount of shift required to correctly compute weight de/potentiation, as shown in **Figure 3**, where the black part shows the spike timing and traces using the presynaptic spike time as the reference, while the red part shows how this reference is shifted once delay has been reintroduced. Not considering the delay generates substantial errors in the weight update. A pseudocode version of the algorithm is presented in Appendix B.


A

Plasticity event (every 128 ms)
– post-synaptic traces are computed
– a walk of the weight matrix is initiated

Packet received:
– spike timing of the post-neuron is stored

Memory Received:
– next synaptic row is requested
– pre-synaptic trace is computed from the spike-history bitmap stored in SDRAM
– LTP is calculated using the value of the pre trace at post spike timings
– LTD is calculated using the value of the post trace at pre spike timings
– synaptic row is stored back to SDRAM

FIGURE 3 | (A) Algorithm for STDP learning implementation on the plastic core **(B)** STDP function **(C)** Implementation of pair-based STDP with local traces and delays, as suggested by Morrison et al. (2008): potentiation occurs at post-synaptic spike times and corresponds to the value of the pre-synaptic trace; conversely, depression happens at pre-synaptic spike times and corresponds to the value of the post-synaptic trace. *d* represents the delay, reintroduced at the post-synaptic end; black and red lines represent the traces and spike timings when the delay is reintroduced (red) as opposed to using the presynaptic spike time as reference (black).

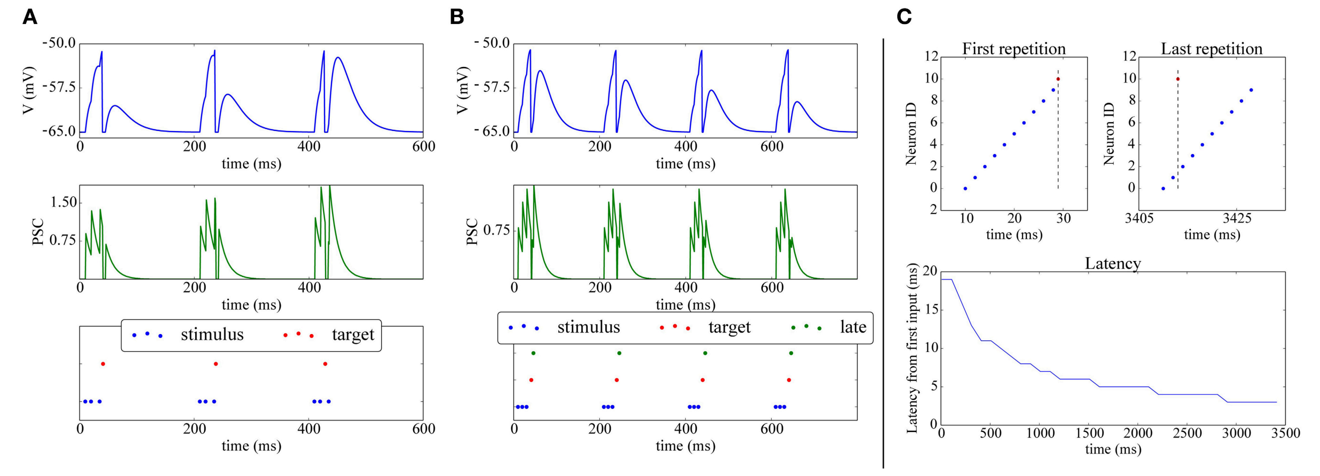


FIGURE 4 | Shift of post-synaptic firing onset via STDP. (A) Potentiation: The spike raster plot (bottom) shows that at the beginning of the stimulation 3 input spikes (blue) are needed to make the target neuron (red) fire; after 400ms, potentiation has made the synapse strong enough so that the post-synaptic neuron fires after only 2 spikes. This is also visible in the membrane potential (top) and post-synaptic currents (PSC; middle). **(B)** Depression: (bottom) The green neuron is made to spike consistently after the target (red) neuron, hence its weight gets depressed, as can be observed by its decreasing contribution in the membrane potential (top) and PSC (middle). **(C)** Reduced spike latency: at the beginning of the simulation (upper-left panel) 10 spikes from 10 different input neurons (blue) are needed to make the post-synaptic (red) neuron fire; after repeated stimulation (upper-right-panel), potentiation via STDP makes the red neuron fire already after 2 spikes, hence latency firing closer to the pattern's start, which is also shown by the latency plot (bottom panel).

A simple experiment in which STDP, implemented with the above scheme achieves synaptic potentiation and depression is shown in **Figures 4A,B**. The final part of the Figure presents a classical experiment where a plastic neuron can reduce the latency of its firing to a repeatedly presented pattern (Mehta et al., 2000; Song et al., 2000): a (red) neuron receives connections from 10 inputs neurons (blue) which fire at 2 ms from each other; during the first repetition all the 10 input neurons are required to make the target neuron fire. After repeated presentations, due to potentiation, only two input neuron spikes are needed to elicit activity in the post neuron, which responds with a lower latency to the onset of the pattern.

4.2. RESULTS: PRE-POST PAIRING USING A TEACHER SIGNAL

In **Figure 5** we reproduce results of a classical stimulation protocol for potentiation induced by pre-post synaptic pairing. The network comprises a *stimulus* population and a *target* population, each separately driven by two different Poisson sources emitting spike bursts at high frequency (350 Hz) for short periods of time (20 ms). Both populations also receive independent background noise. The Poisson and noise source populations are interconnected with a one-to-one connectivity pattern to their respective inputs and outputs. The stimulus and the target populations are interconnected with a 50% probability.

At the beginning of the simulation, external stimulation coming from the stimulus population is not strong enough to trigger activity in the target post-synaptic population ($0 \leq t \leq 1500$ ms). Afterward ($1500 \leq t \leq 3000$ ms) the stimulus and target populations are stimulated together by their respective Poisson inputs, so that the target population spikes 10 ms after the stimulus population, hence inducing potentiation. Finally, for $3500 \leq t \leq 4000$ ms, the Poisson process feeding the post-synaptic population is removed, and the post-synaptic population is only stimulated by inputs from the pre-synaptic population. It can be seen that because of the induced potentiation, the pre-synaptic input is now strong enough to make the target population fire without any supervisor input.

4.3. RESULTS: BALANCED EXCITATION

Song et al. (2000) have shown that STDP can establish a state of balanced excitation in the post-synaptic neuron, which makes it more likely to fire with a controled output rate in response to fluctuations in its input. This is achieved by competition between the synapses that project onto the post-synaptic neuron, induced by STDP. The characteristic effect described by Song et al. (2000) is that STDP creates a bimodal distribution of input weights, pushing them either toward the minimum or maximum values, and creating groups of *strong* and *weak* synapses. In **Figure 6** we simulate a group of 1000 input neurons, firing independently according to a Poisson process at 20 Hz, and projecting onto a single output neuron. The weights are initialized uniformly, and then undergo STDP. After 300 s of simulation, the distribution of synaptic weights in **Figure 6** shows clearly the characteristic separation into two groups of very different strengths. The experiment can be observed in Movie 1 (Supplementary Material), which shows the weight distribution as the simulation is running.

5. BCM

The *BCM rule*, named after their inventors Bienenstock, Cooper, and Munro (Bienenstock et al., 1982), is a rate-based synaptic plasticity rule, introduced to model binocular interactions and the development of orientation selectivity in neurons of the primary visual cortex. The BCM rule is based on Hebbian principles, but introduces synaptic competition by correlating

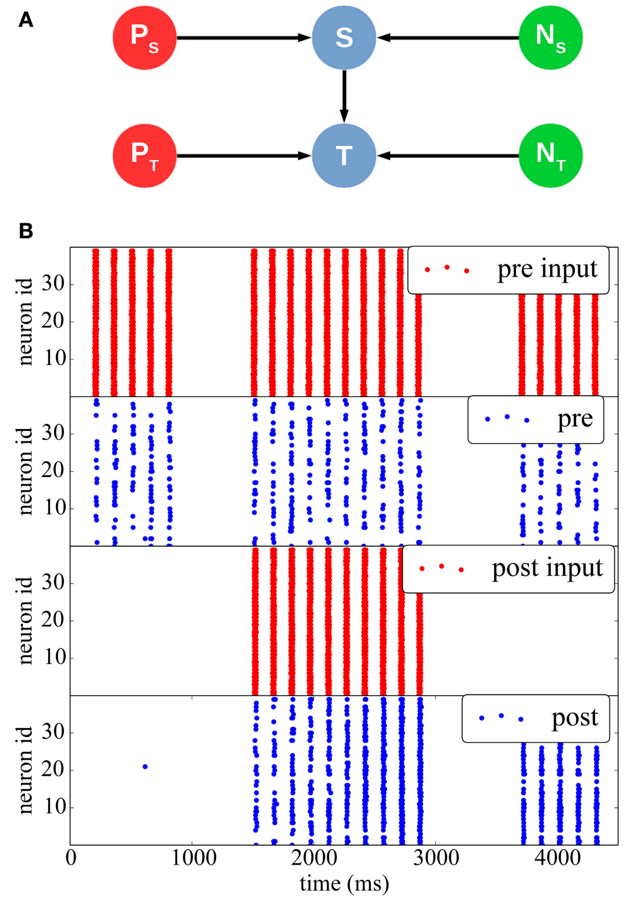


FIGURE 5 | STDP with a teacher signal. (A) Network structure: two different Poisson spike sources (red) are used as supervisor signals to individually stimulate the *Stimulus* (top) and *Target* (bottom) populations at different times (blue), which also received noise from two separate sources (green). **(B)** Initially (between 0 and 1000 ms), only the pre-synaptic population is stimulated, but the synaptic weights are weak, thus the resulting spikes (blue) do not elicit post-synaptic spikes. Between 1500 and 3000 ms, both populations are stimulated with a 10 ms time difference, such as to induce synaptic potentiation. The effect can be seen between 3500 and 4500 ms, when the teacher signal for the post-synaptic population is removed: after the potentiation the pre-synaptic spikes are able to drive the post-synaptic neurons by themselves.

the pre-synaptic rate with a non-linear function of the post-synaptic rate. In its simplest form the BCM rule computes this non-linearity as the product of the post-rate with its deviation from the mean post-synaptic activity (see **Figure 7B**):

$$\frac{dw}{dt} = [r_{post}(r_{post} - \theta)r_{pre}]\delta - \epsilon \cdot w \quad . \tag{4}$$

Here w denotes the synaptic weight and dw/dt its change, r_{post} and r_{pre} are the firing rates of the pre- and post-synaptic neurons, θ is the modification threshold, which is computed here as the mean firing rate of the post-synaptic neuron, δ is a learning rate, and ϵ a weight-decay parameter. If r_{post} exceeds the mean firing rate θ, the weight is potentiated; conversely, for lower activity ($r_{post} < \theta$) the weight is depressed. The learning rate parameter δ can be used to normalize the magnitude of the synaptic weight change according to the neural model used. Many variations of the BCM rule have been studied since its introduction, using different kinds of non-linearities, but here we study only the basic version from Bienenstock et al. (1982).

5.1. METHODS: IMPLEMENTATION OF BCM ON THE PLASTICITY CORE

Since the BCM rule only requires firing rates, the plasticity core just has to increment a counter whenever a post-spike is received, and to use a low-pass filtered version of the rate. Analogously, when processing a row relative to an afferent neuron, the number of spikes received during the previous phase is used to update the pre-synaptic rate information. At the end of each plasticity phase θ (the threshold parameter representing the mean rate) is updated using a configurable exponential moving average, and the pre spike windows are reinitialized. A pseudocode version of the algorithm is presented in Appendix C and is outlined in **Figure 7A**.

In **Figure 7C** we show a classical potentiation protocol using the BCM rule. For the first 600 ms the target population is only receiving spikes from the stimulus population, but the weights are too weak to cause firing in the target population. Between 600 and 1200 ms, a teacher population is activated which is strong enough to drive the target population, thereby potentiating also the simultaneously active stimulus-target connections. Afterwards, when the teacher population is switched off, the stimulus population alone is able to drive the target population without teacher input.

5.2. RESULTS: EMERGENCE OF ORIENTATION SELECTIVITY WITH BCM

The BCM rule has been originally proposed in Bienenstock et al. (1982) to explain how neurons in the primary visual cortex can acquire their feature selectivity from sensory stimulation. As a test of our implementation of BCM on SpiNNaker we replicate a simple neural network with lateral inhibition which undergoes plasticity while receiving monocular visual input in the form of oriented bars.

The network consists of 2 layers, an input layer which comprises 16 × 16 neurons and an output layer with 4 neurons. Each neuron in the input layer projects, in an all-to-all fashion, to the output neurons. All synapses are initialized with random weights and delays. Each neuron in the output layer has an inhibitory projection to every other neuron, forming a network of lateral antagonism (Shouval et al., 1997). The aforementioned connectivity pattern matches anatomical data, for example the *lateral plexus* of the Limulus's eye, as originally found by Hartline et al. (1956). Bienenstock et al. (1982) themselves point out that no selectivity is achieved without lateral inhibition.

For this experiment four images of oriented bars are used as input stimuli, each rotated by 45°C. Bars are 3 pixels thick and 12 pixels in length, and the intensity of each pixel is a random value between 0.8 and 1.0. Each pixel is converted to Poisson spike trains, in order to simulate spikes coming from the retina or LGN. The firing rates are proportional to the value of the pixels, while all firing rates are scaled such that the input layer generates approximately 1000 spikes per second. During the simulation each orientated bar is presented to the network in a

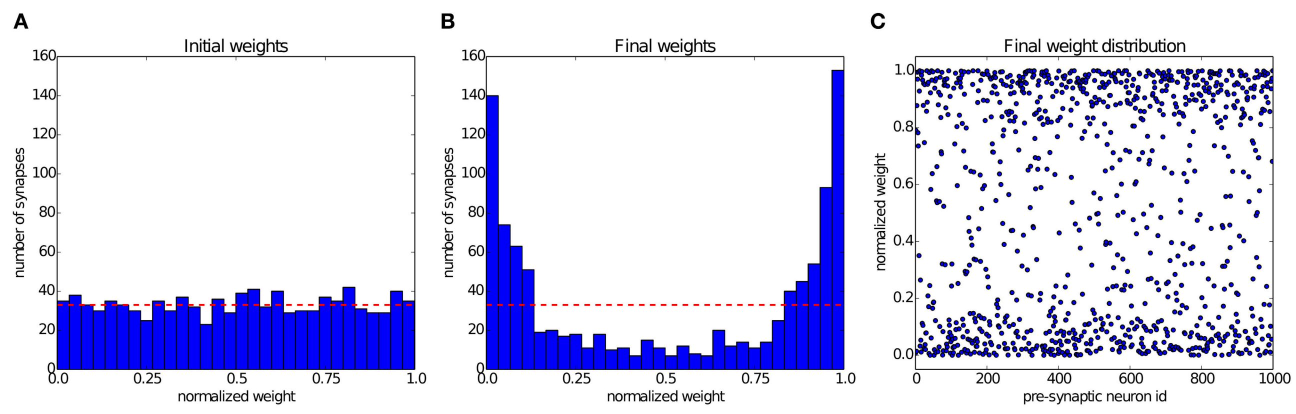


FIGURE 6 | Competition between synapses undergoing STDP: In the experiment introduced by Song et al. (2000), 1000 uncorrelated pre-synaptic neurons, firing at a Poisson-rate of 20 Hz, project onto a single post-synaptic neuron. (A) Initial uniform weight distribution before plasticity. **(B)** After 300 seconds of stimulation STDP has divided the synaptic weights into *weak* and *strong* ones, thereby regulating the activity of the post-synaptic neuron. The red line shows the mean of the initial weights. **(C)** Scatter plot of the final weight distribution.

A

Plasticity event (every 128 ms)
– a walk of the weight matrix is initiated
– theta is updated using a configurable moving exponential average

Packet received:
– spike count of the post-neuron is incremented and used as a rate

Memory Received:
– next synaptic row is requested
– rate of the pre neuron is computed from the spike history bitmap stored in SDRAM
– rate part of the weight rule is applied
– weight relaxation is applied
– synaptic row is stored back to SDRAM

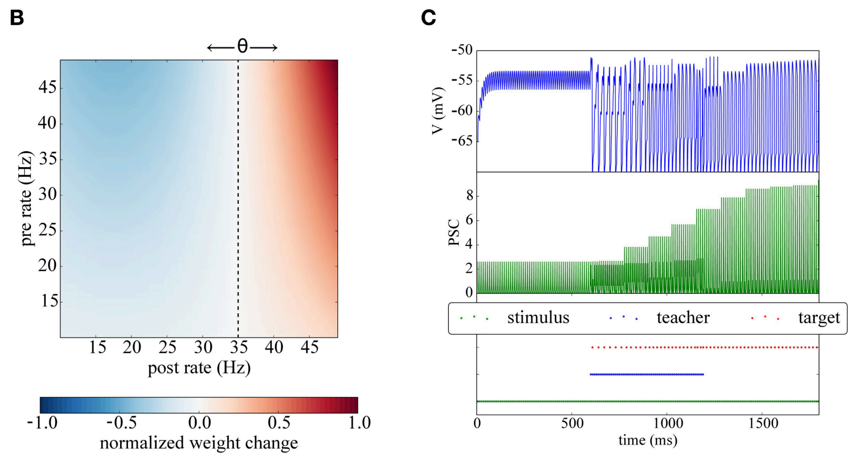


FIGURE 7 | Implementation of BCM plasticity. (A) Algorithm for BCM learning on the plasticity core. **(B)** Illustration of the BCM rule: normalized weight change as a function of the pre- and post-rates, with $\theta = 35$ Hz. **(C)** Potentiation experiment using a teaching signal: when stimulation is paired with a teacher signal that forces the post-synaptic neurons in the target population to fire, the weights get potentiated and become strong enough to drive the post-synaptic neuron.

random order for 1 s and for 80 repetitions. Learning takes place in the synapses between the input and output layer, while the inhibitory synapses in the output layer are static and set to a weight of −9 nA.

The results are summarized in **Figure 8**. **Figure 8A** shows that the weights and neuronal responses to input stimuli are initially random. At the end of the simulation, **Figure 8B** shows that each output neuron has developed via BCM plasticity a receptive field that corresponds to one particular orientation. In **Figure 8C** we show the orientation tuning curves of each neuron, measured by rotating the stimulus bar counter-clockwise in 10°C steps. The results show that each neuron has successfully learned to respond best to one preferred orientation, which is in line with previous modeling studies and experimental and anatomical data (Moore and Freeman, 2012; Jeyabalaratnam et al., 2013). The learning can be observed in Movie 2 (Supplementary Material), where the four receptive fields are emerge from the repeated presentation of the input stimulation.

6. VOLTAGE-GATED STDP

Brader et al. (2007) have presented an STDP rule that is triggered by the arrival of pre-synaptic spikes, and in which the change in synaptic efficacy is a function of post-synaptic depolarization and of an internal variable at the spike arrival time. The rule is motivated by the necessity to design learning rules which are at the same time biologically plausible, but also compatible with implementation constraints on neuromorphic devices. Several studies have demonstrated the ability of the learning rule to discriminate complex spatio-temporal patterns (Indiveri and Fusi, 2007; Giulioni et al., 2009; Mitra et al., 2009), even if the synapses are allowed to take on only one of two stable states. Every time the post-synaptic neuron emits a spike an internal variable $C(t)$,

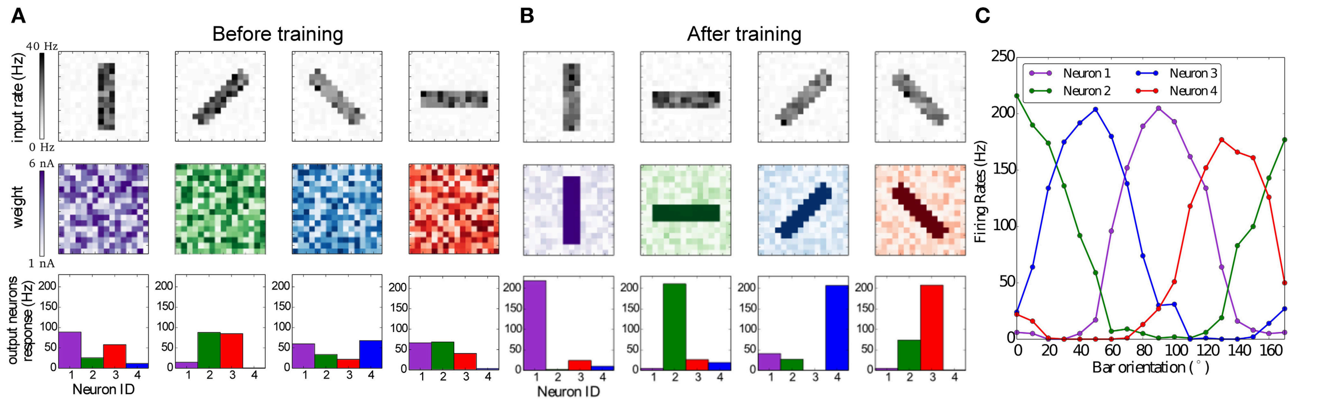


FIGURE 8 | Emergence of orientation selectivity with the BCM learning rule. (A,B): The top row represents the input stimuli bars presented in different orientations, the total firing rate for each stimulus is 1,000 Hz; the middle row shows the weight matrix for the output neurons, with **(A)** random initial weights and **(B)** results after the training, where it can be observed that neurons have developed their receptive fields according to the input stimulation; the bottom row shows the firing rates of the output neurons, where each color codes for a different neuron which has learned a preferred orientation. **(C)** Orientation tuning curves obtained by rotating a horizontal bar counter-clockwise with a step size of 10°C.

representing calcium concentration due to back-propagating action potentials, is incremented by a value J_C and then decays with a time constant τ_C according to the dynamics described by

$$\frac{dC(t)}{dt} = -\frac{C(t)}{\tau_C} + J_C \sum_i \delta(t - t_i), \quad (5)$$

where t_i are the post-synaptic spike times. Potentiation and depression happen only if $C(t)$ is in an appropriate interval $[\theta^h_{down}, \theta^h_{up}]$ for potentiation and $[\theta^l_{down}, \theta^l_{up}]$ for depression. Post-synaptic membrane depolarization $V(t)$ influences this plasticity rule, triggering potentiation (or depression) only if the membrane potential of the post-synaptic neurons is higher (lower) than a threshold value θ_V at the time of arrival of a pre-synaptic spike (t_{pre}). Modification of the synaptic efficacy w can then be summarized by the following equations:

$$w = w + a \text{ if } V(t_{pre}) > \theta_V \text{ and } \theta^h_{down} \le C(t_{pre}) < \theta^h_{up} \quad (6)$$

$$w = w - b \text{ if } V(t_{pre}) \le \theta_V \text{ and } \theta^l_{down} \le C(t_{pre}) < \theta^l_{up} \quad (7)$$

where a and b represent the constant weight increase and decrease values respectively.

If none of the conditions in (6) and (7) are met, or if no spike is received in the period of time considered, then the weight drifts toward one of two stable values (w_{min} and w_{max}). The direction of the drift is determined by comparing the current weight w to a threshold θ_W, and speed of the drift toward the minimum and maximum stable states is determined by the constants α and β respectively. This leads to the following dynamics:

$$\frac{dw(t)}{dt} = \alpha \quad \text{if} \quad w(t) > \theta_W \quad (8)$$

$$\frac{dw(t)}{dt} = -\beta \quad \text{if} \quad w(t) \le \theta_W \quad (9)$$

6.1. METHODS: IMPLEMENTATION OF VOLTAGE-GATED STDP ON THE PLASTICITY CORE

The voltage-gated STDP rule needs further information from the post-synaptic neuron, as the membrane potential gates potentiation or depression. The cores communicate this information as means of shared memory in SDRAM, using a double buffer technique so that they always work on different phases. This induces a slight overhead in the neural core, which has to perform the check against θ_V and saves the result for each millisecond in a bitmap stored in memory. The plasticity core retrieves the results of the comparison at the beginning of each plasticity phase, and uses them in the weight update process. At the same time the function $C(t)$ is computed starting from the post neuron spike timings, similarly to computing the STDP traces. A pseudocode version of the algorithm is presented in Appendix D.

The basic dynamics of this voltage-gated STDP rule are shown in **Figure 9**: The bottom row shows the trace of the calcium variable $V(t)$, which is increased by J_C whenever the post-synaptic neuron fires, and then exponentially decays with time constant τ_m. The central part shows the potentiation of a synapse, because here the pre-spikes arrive when $V(t_{pre}) > \theta_V$ and $\theta^h_{down} \le C(t_{pre}) < \theta^h_{up}$, and thus fewer spikes are needed to drive the target neuron. Conversely, on the right we observe the depression of a synapse, because pre-spikes arrive when $V(t_{pre}) \le \theta_V$ and $\theta^l_{down} \le C(t_{pre}) < \theta^l_{up}$. After depression, the synaptic input is too weak to make the target neuron fire.

6.2. RESULTS: LEARNING TEMPORAL PATTERNS

To verify our implementation of the voltage-gated STDP rule by Brader et al. (2007), we implemented the model by Coath et al. (2013) for learning temporal structures in auditory data, which has originally been implemented on a neuromorphic chip in Sheik et al. (2012). The study focused on learning dynamical patterns in the context of a sound perception model by tuning auditory features through presentation of stimuli and learning using the STDP rule implemented in VLSI.

A

Plasticity event (every 128 ms)
– the C(t) trace is computed from the spike history bitmap stored in DTCM
– membrane potential threshold checks are retrieved from a shared portion of SDRAM
– a walk of the weight matrix is initiated

Packet received:
– spike time of the post-neuron is stored

Memory Received:
– next synaptic row is requested
– weights are updated accordingly to the rule
– weight relaxation towards one of two states is applied
– synaptic row is stored back to SDRAM

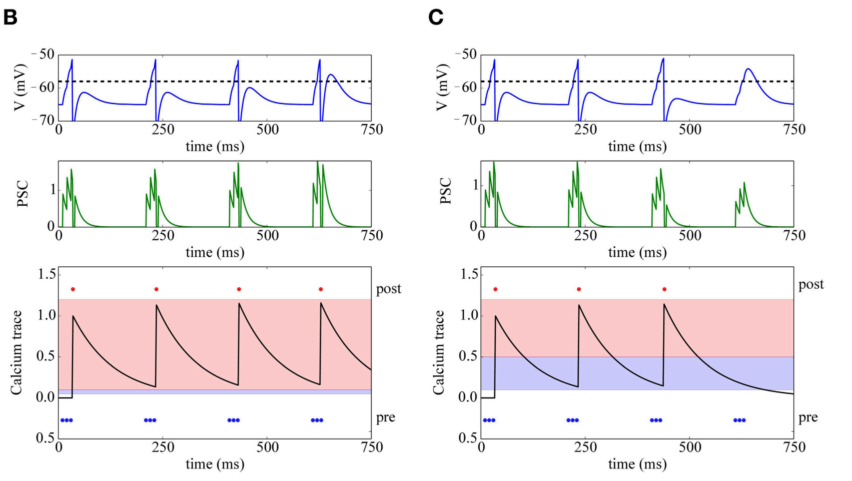


FIGURE 9 | Implementation of voltage-gated STDP. (A) Concept for implementing voltage-gated STDP on the plasticity core. **(B)** Example of synaptic potentiation: three pre-synaptic spikes (blue) arrive while the membrane potential is greater than θ_V and $\theta^h_{down} \leq C(t_{pre}) < \theta^h_{up}$ (red-shaded area in bottom row). Initially, the post-synaptic neuron (red) fires after the third spike, after potentiation only two spikes are needed to make the target neuron fire. **(C)** Depression example: three pre-synaptic spikes (blue) arrive while the membrane potential is less than θ_V and $\theta^l_{down} \leq C(t_{pre}) < \theta^l_{up}$ (blue-shaded area in bottom row). After depression takes place, the post-synaptic neuron (red) no longer fires after receiving the 3 input spikes.

The proposed network learns to respond to particular input timing patterns. The network comprises 3 layers of tonotopically organized frequency channels, representing different positions on the basilar membrane. The first layer A represents a spiking signal produced by an artificial cochlea (such as the one in van Schaik and Liu, 2005); each neuron in the A layer projects to a neuron in 2 layers, B_1 and B_2 through excitatory synapses, while B_1 projects to B_2 through inhibitory synapses. Each neuron in B_2 also receives plastic connections from all the neighboring B_1 neurons, with delays proportional to the distance, as shown in **Figure 10**. Since delays are programmable in SpiNNaker we incorporated them directly in the B_1 to B_2 connection, and not through a separate neural population as in the original model. This delay property is essential for learning: correlation between the delayed feedback arriving from other B_1 neurons to the B_2 neurons is detected by the plasticity rules, and it controls synaptic potentiation and depression by coincidence detection. To implement the model on SpiNNaker while coping with the 1 ms time resolution used in the current neural kernels we multiplied all the time quantities by 10. For learning we use the same three input patterns that were used in the original model (see **Figure 10**): Pattern (C) is a forward frequency sweep, where every frequency (and therefore every A neuron) is activated in order, with a short delay between one presentation and the next. For pattern (D) we perform the same frequency sweep, but we move backwards through the frequency space. Finally for pattern (E) we perform a forked frequency sweep, starting from the middle frequency. We present the stimulus multiple times to the network and analyze what it has learned by examining the B_1/B_2 weight matrix. The results are presented in **Figure 10**, and can be compared with the results in Figures 7, 8 in Sheik et al. (2012). After repeated presentations of the target patterns, the weight matrix, initialized randomly, converges to a state where it is only sensitive to the spike-timing pattern presented, by coincidence detection through delay lines. The emergence of the connectivity matrix for the forked frequency sweep can be observed in Movie 3 (Supplementary Material).

7. PERFORMANCE ANALYSIS AND DISCUSSION

In Diehl and Cook (2014), the authors describe an STDP variation of the DED which follows the strategy proposed in Morrison et al. (2008) by storing traces in SDRAM, rather than performing spike pairing as proposed in Jin et al. (2010b). The authors evaluate the performance of their implementation as well as the one present in the stable release of the SpiNNaker software package[1] in terms of synaptic events processed per second, as done in Sharp and Furber (2013) and Stromatias et al. (2013). They do so by feeding a leaky integrate-and-fire population of 50 neurons with a neural population of variable size that produces spikes at ≈250 Hz, according to a Poisson process, with a 20% connection probability. They report that their implementation of plasticity is capable of handling around 500K synaptic events per second per core (using 150 input neurons), while the original SpiNNaker implementation is limited to 50K events.

We adopt a similar strategy to evaluate our plasticity algorithms, but in more stringent conditions, and with a larger connectivity range. Rather than testing a single core we test a full chip (16 cores). In this way, we can also evaluate the effects of memory contention between different cores, as memory access can be a bottleneck for simulations on SpiNNaker. We model a population of 800 neurons in a single SpiNNaker chip (8 cores modeling neurons and 8 cores dedicated to plasticity) fed by an input Poisson neural population of 150 neurons with a variable rate, and measure the maximum firing rate at which the simulation can run in real time. We take as a starting point the connectivity levels reported by Diehl and Cook (2014) (20%

[1]https://spinnaker.cs.man.ac.uk/tiki-index.php?page=SpiNNaker+Package+(quarantotto)

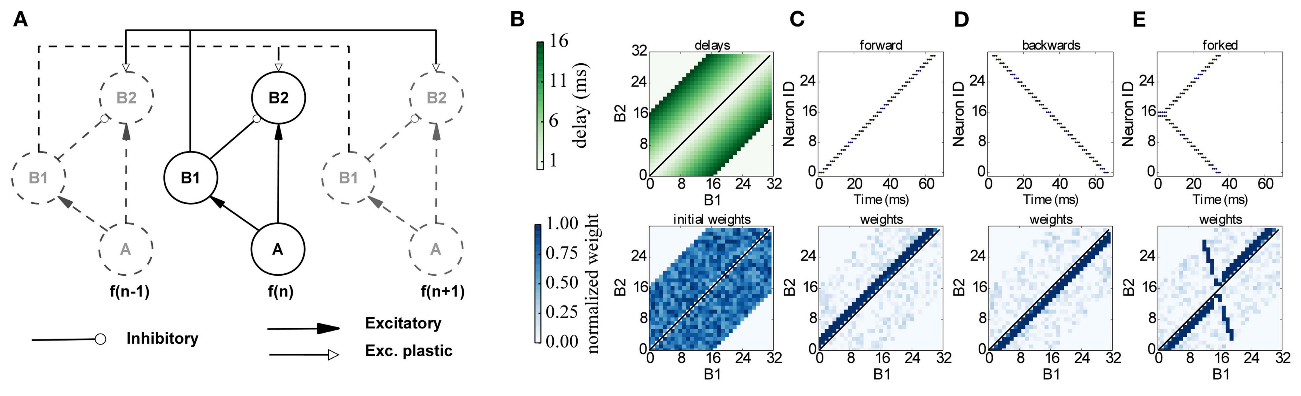


FIGURE 10 | Learning temporal patterns with the model by Coath et al. (2013) and Sheik et al. (2012). (A) Network structure: each frequency channel *f* comprises three neurons *A*, B_1 and B_2, and is connected to other frequency channels (dashed in figure), arranged in a tonotopical way, with distance-based delays and plastic connections. **(B)** Delay matrix (top) and example of initial random weight matrix (bottom). **(C–E)** Input spikes (top) and resulting weight matrix (bottom) after learning for a forward, backwards and forked frequency swipe respectively.

interconnection probability, 150 × 50 × 0.2 synapses, for a total of 1,500 per core and 24,000 per chip if considering 16 cores) and increase the connectivity level up to 100% (7,500 synapses per core, 120,000 per chip). This results in synaptic rows which are 5 times longer, as every pre-synpatic neuron is connected to every post-synaptic neuron in each core, rather than only 20% as in the original experiment. We then scale the model further up by adding more pre-synaptic neurons so as to reach a total of 156,000 synapses. The performance analysis of the algorithms proposed in this work uses the same leaky integrate-and-fire current based neuron. To be able to scale the rate while maintaining the post-synaptic activity constant, we set all the weights and all the weight increments in the plasticity rules to 0, similarly to the approach in Stromatias et al. (2013). This means that plasticity is normally computed, but the weight is clipped to 0 and stored back in SDRAM. Such values are set at runtime and cannot therefore by optimized by the compiler; we have also ensured that setting these values to 0 would not bypass part of the code by removing some optimization tests (like not updating weights which do not change), thus ensuring that the code behaves in our test case as the worst possible real case. Post-synaptic activity is induced by feeding the leaky integrate-and-fire neurons with a current inducing an activity of ≈ 22 Hz.

We check if at any moment any core is lagging behind real time as this would make the simulation incorrect and unrepeatable. We also check if a walk through of the weight matrix is completed before the end of the plasticity period or, in other words, if the plasticity process is finished before the next one starts, as overlapping in this sense is not possible when operating in real time. This allows us to measure the maximum number of synaptic events that can be handled in real time by a single SpiNNaker chip, using the three learning rules proposed in this paper (STDP, BCM and voltage-gated STDP), and to understand if the performance is limited by the neural or the plasticity core.

Results are shown in **Figure 11**; for each given connectivity level (number of synapses) pre-synaptic firing rates are increased until the limit after which real-time simulation is no more possible. Each point of the plot hence represents the limits of the approach for a given connectivity, for each of the plasticity rules implemented. From the Figure it can be observed that the three learning rules implemented within this framework have similar performances untill the limit of 96,000 synapses (corresponding to scaling up to 80% connectivity the model by Diehl and Cook, 2014). This is due to the fact that, up to that point, all three learning rules are limited by the neural cores lagging behind real time, rather than by the plasticity process taking too long. Such limit peaks just below 1,5 million synaptic events per second per core for all three rules (23 million events for the full chip). In a non-plastic performance analysis, Stromatias et al. (2013) measured a maximum throughput of ≈ 2.38 million synaptic events per second per core. After this connectivity level the complexity of the two STDP models (standard and voltage-gated) becomes the limiting factor, and a complete walk of the synaptic matrix is not possible anymore within the 128 ms period used in this paper. The BCM algorithm is not affected by this, as the algorithm is computationally less intense, and keeps improving above 1,6 M synaptic events per second per core. The decay in performances reflects the complexity of the algorithm considered: standard STDP, being more complex than the voltage-gated version, has a sharper decrease in performances.

When comparing these scenario results with the previous plasticity models based on the DED by Jin et al. (2010b) and Diehl and Cook (2014) (around 50k and 500k synaptic events per second per core respectively in the 20% case - the leftmost part of **Figure 11**), it must be remembered that these algorithms work with a 1 ms spike-window resolution, while the experiments proposed in this paper have adopted a resolution of 2 ms. Also the former algorithms might lose spikes, while in the approach presented here the contributions from *all* the spikes are accumulated (or, in other words, no spike is lost).

While our approach was designed for maximal flexibility, there might be tradeoffs in terms of efficiency for some scenarios, depending on connectivity and firing rates. One limitation of our approach is, for example, that every plasticity event triggers an update of the complete synaptic matrix. For the rules proposed in this paper is not possible to selectively update only some

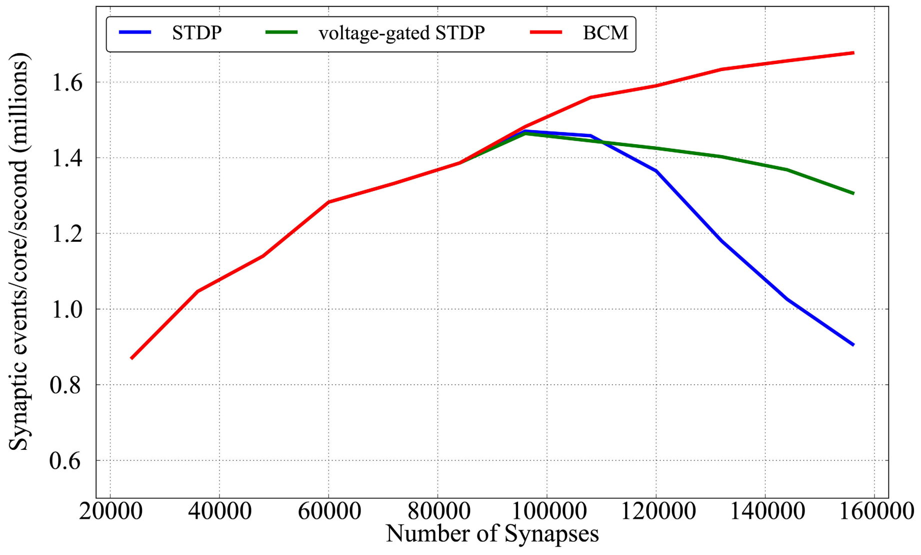


FIGURE 11 | Performance evaluation of the three learning rules in terms of synaptic event processed per core per second as a function of different number of synapses.

rows. For pre/post sensitive rules (such as STDP) destinations are encoded in the synaptic row, which is stored in SDRAM, so it is not possible to know if a pre neuron connects to a post neuron which has fired (thus inducing LTP) before retrieving the row itself. In rate based models such as BCM, where the firing rate is considered as a moving average, the absence of spikes is not sufficient to ensure there is no plasticity in act. Finally rules with relaxation toward one (BCM) or multiple (voltage-gated STDP) states require a weight update even in the absence of a spike.

Since in our implementation plasticity updates occur every 128 ms, pre-synaptic firing rates should be at least on the order of $\approx 7 - 8$ Hz to avoid having to update silent synapses regularly. In scenarios with lower firing rates, a purely event-driven update would be more efficient. However, a main motivation for our approach is to ensure real-time performance, even in situations with momentarily high load, e.g., if multiple neurons are firing in bursts. Such scenarios are common when using natural inputs with coincident input spikes or models with oscillatory background signals. In such cases the plasticity core approach offers greater flexibility to process plasticity in real time: instead of having to process neural and synaptic updates of all simultaneous spikes within the 1ms time step of the neural core, which might be challenging for complex plasticity rules or for complex neural models, our approach accumulates events over the longer time window of the plasticity core.

This decoupling enables the neural cores to maintain the real-time constraints, and opens up new possibilities for trade-offs to reduce the load on the plasticity cores if necessary. The simplest possibility is, as in the DED model, to lower the number of neurons simulated by each neural core (and therefore also by its associated plasticity core). Other options, although not implemented in the first proof-of-concept presented in this paper, are possible. For the initial results presented in this work we maintain a 1:1 ratio between neural and plasticity cores, but this will likely not be optimal for all scenarios. When looking at **Figure 11** it can be see that the two STDP models show a sweet spot for performance at around 80% connectivity. Before such maximum the performance is limited by the neural cores, while after that is the plasticity core which is not able to keep up with the real-time requirements. An interesting alternative would be to allocate more plasticity cores to a single neural core, and adapt the *plasticity:neural* core ratio according to the network characteristics and to the computational complexity of the neural and plasticity algorithms and the associated workload.

A limiting hardware factor for any implementation of plasticity on SpiNNaker is the memory bandwidth, because rows of the synaptic matrix need to be written back to SDRAM. It was shown in Figure 9 of Painkras et al. (2013) that writing is the main bottleneck, since the read bandwidth is twice as high. Our approach reduces the write load, since rows are only written back to SDRAM at most once every plasticity interval, rather than once every pre-synaptic spike as in the DED model. This means that, for example, if pre-synaptic neurons are firing at 24 Hz each synaptic row would be transferred back to SDRAM 24 times per second using the DED model, but only 8 times with our approach.

Finally another possibility is to increase the duration of plasticity intervals, which increases the time available for computing the updates, but comes at the cost of larger memory requirements for storing traces in the core-local DTCM. For long plasticity intervals this might grow beyond what can be stored in DTCM (64 Kbytes for each ARM core, of which some space needs to be reserved for other parameters and buffers). The capacity can be increased by lowering the precision for storing the traces, or using a coarser time resolution. All these possible trade-offs, although not fully explored in this initial work, show the versatility of the approach, which can be adapted to different situations and modeling needs, and constitutes one of its key features, as discussed in the last Section.

8. DISCUSSION

Current research on understanding the relationship between the local electrochemical processes of synaptic plasticity and their manifestations as high-level behavioral learning and memory is increasingly relying on theoretical modeling and computer simulations (Gerstner et al., 2012). Because of the great diversity of plasticity phenomena observed in biology and the resulting diversity of proposed mathematical models, as well as the computational complexity of spiking neural network simulations dominated by the costs of synaptic processing, it is necessary to create simulations tools that provide both the flexibility to try out new models easily, and the speedup of specialized hardware. This meets the demand of increasingly large neural network simulations, both for studying brain function, and for applications in artificial intelligence (Le et al., 2012). SpiNNaker has proven to be a well-suited platform for massively parallel large-scale simulations of spiking neural networks, and is flexible enough to let researchers implement and test their own computational models in standard programming languages. The previous Deferred Event Driven Model of handling events in SpiNNaker has made it difficult to implement plasticity rules with arbitrary triggering events (pre-, or post-synaptic, or at regular time intervals), rules which depend on third factors available only at the post-synaptic neuron, or plasticity in networks with variable axonal delays. We have presented here a framework which uses the modular architecture of SpiNNaker and delegates weight updates to dedicated plasticity cores, while the network simulation operates on the remaining neural cores. We have shown that a variety of commonly used plasticity rules can be exactly replicated on this framework, with a greatly increased capacity of processing plasticity events in real-time, by running experiments on a 4-chip SpiNNaker board. The separation of neural from plastic concerns is the feature that enables the great flexibility of the architecture. The two cores work in parallel on different time scales and phases, and the plasticity core has all the information to compute plasticity for the recent past, can access the weight matrix shared with the neural core, and any other information that can be passed through means of shared memory, e.g., membrane potentials and spike timings of the pre- and post-synaptic neurons. All this information can be pre-processed before plasticity is computed, which allows e.g., the computation of rates in an otherwise spike-based simulation. The architecture can be configured easily, using PyNN scripts. This standard, high-level neural language makes it easy to integrate and explore new learning rules into the SpiNNaker architecture.

The approach presented in this paper is tailored to SpiNNaker and to its specific architecture, design and constraints. Nonetheless the same principles could be applied to other digital-analog hybrid architectures, where efficient neural simulation could be realized on one neuromorphic chip, whereas complex plasticity rules could be realized off-chip on computers or FPGAs. Regarding GPUs it appears to be more favorable to let each kernel perform the same operation following the SIMD paradigm. Fidjeland et al. (2009) sequentially use two different kernels, one for neural updates and one for applying plasticity updates. Such kernels do not run in parallel on the same GPU, but serially. This does not constitute a problem when running accelerated simulations, which is the common case for GPUs, but can raise difficulties when running in closed-loop real-time scenarios, as in neurally inspired robotics (Galluppi et al., 2014). In fact concurrent kernel execution is a feature that has only recently been introduced in GPUs, with the NVIDIA Fermi architecture. Using such technique, a plasticity and a neural kernel could be instantiated concurrently on the same GPU, in a similar way to what is done in our approach. Memory access patterns, and the possibility of accessing contiguous portions of memory is a key factor when programming a GPU (Brette and Goodman, 2012). It could be speculated that applying an approach like the one proposed in this paper would have the benefit of guaranteeing memory coalescence, as the synaptic matrix is sequentially accessed when walking through it. Multi-core or cluster architectures could also in theory benefit of separating neural simulation and plasticity, running either on different threads or on different cores, and with different time scales. However, clusters are equipped with more powerful processing units than SpiNNaker, so computing neural and synaptic updates in different cores could introduce unnecessary overheads and synchronization difficulties, particularly regarding memory bandwidth and access patterns.

In our experiments we have deliberately chosen to reproduce classical results, in order to compare the run-time performance of the novel framework to previous implementations of plasticity on SpiNNaker. The examples of BCM, STDP, and voltage-gated STDP learning provide templates for constructing further experiments with rate-based, spike-timing-based, and voltage-dependent learning rules. Our approach can be easily extended to include additional third factors to modulate plasticity, e.g., neuromodulators (Izhikevich, 2007; Potjans et al., 2011), or weight-dependency (Morrison et al., 2008; Nessler et al., 2013), can model homeostatic effects (Bartolozzi et al., 2008), or handle different synaptic delays (Tapson et al., 2013; Wang et al., 2013). It can also combine different models of plasticity in one simulation, a feature which is used in several recent models, where network function arises from the interaction of different synaptic plasticity rules that are specific to particular cell types (Lazar et al., 2009; Savin et al., 2010; Binas et al., 2014; Kleberg et al., 2014). In fact, we have provided a tool that should be general enough to model long-term potentiation rules, but is not restricted only to phenomenological ones. Other biological structures i.e., glial cells are considered to have a fundamental role in plasticity, and can enhance learning capabilities (Min et al., 2012). The plasticity core, by leveraging this functional segregation already present in biology, is a natural candidate to model such structures.

The results presented in this work and the possibilities opened by this approach point to the efficiency and to the generality of the framework introduced: a modular, flexible and scalable tool for the fast and easy exploration of learning models of very different kinds on the parallel SpiNNaker system.

ACKNOWLEDGMENTS

This work has been partially supported by Labex Livesenses (Eventsee) and by the Human Brain Project (aCORE). The

SpiNNaker project is supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/G015740/01. Michael Pfeiffer has been supported by the Swiss National Science Foundation Grant 200021_146608 and the Samsung Advanced Institute of Technology. The work proposed in this paper is resulted from discussions at the Capocaccia and Telluride Workshops; the authors would like to thank the sponsors and the organizers.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2014.00429/abstract

REFERENCES

Abbott, L. F., and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nat. Neurosci.* 3, 1178–1183. doi: 10.1038/81453

Bartolozzi, C., Nikolayeva, O., and Indiveri, G. (2008). "Implementing homeostatic plasticity in VLSI networks of spiking neurons," in *Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008* (St. Julien's), 682–685.

Beyeler, M., Dutt, N. D., and Krichmar, J. L. (2013). Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule. *Neural Netw.* 48, 109–124. doi: 10.1016/j.neunet.2013.07.012

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

Bienenstock, E. L., Cooper, L. N., and Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *J. Neurosci.* 2, 32–48.

Binas, J., Rutishauser, U., Indiveri, G., and Pfeiffer, M. (2014). Learning and stabilization of winner-take-all dynamics through interacting excitatory and inhibitory plasticity. *Front. Comput. Neurosci.* 8:68. doi: 10.3389/fncom.2014.00068

Brader, J. M., Senn, W., and Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Brette, R., and Goodman, D. F. (2012). Simulating spiking neural networks on gpu. *Netw. Comput. Neural Syst.* 23, 167–182. doi: 10.3109/0954898X.2012.730170

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., et al. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *J. Comput. Neurosci.* 23, 349–398. doi: 10.1007/s10827-007-0038-6

Cassenaer, S., and Laurent, G. (2012). Conditional modulation of spike-timing-dependent plasticity for olfactory learning. *Nature* 482, 47–52. doi: 10.1038/nature10776

Clopath, C., Busing, L., Vasilaki, E., and Gerstner, W. (2010). Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nature Neurosci.* 13, 344–352. doi: 10.1038/nn.2479

Coath, M., Sheik, S., Chicca, E., Indiveri, G., Denham, S. L., and Wennekers, T. (2013). A robust sound perception model suitable for neuromorphic implementation. *Front. Neurosci.* 7:278. doi: 10.3389/fnins.2013.00278

Davies, S., Galluppi, F., Rast, A., and Furber, S. (2012). A forecast-based STDP rule suitable for neuromorphic implementation. *Neural Netw.* 32, 3–14. doi: 10.1016/j.neunet.2012.02.018

Davison, A. A. P., Brüderle, D., Eppler, J. J., Kremkow, J., Muller, E., Pecevski, D., et al. (2008). PyNN: a common interface for neuronal network simulators. *Front. Neuroinformat.* 2:11. doi: 10.3389/neuro.11.011.2008

Diehl, P. U., and Cook, M. (2014). "Efficient implementation of STDP rules on SpiNNaker neuromorphic hardware," in *International Conference on Neural Networks (IJCNN) 2014* (Beijing), 4288–4295.

Fidjeland, A., Roesch, E., Shanahan, M., and Luk, W. (2009). "NeMo: a platform for neural modelling of spiking neurons using GPUs," in *2009 20th IEEE International Conference on Application-Specific Systems, Architectures and Processors* (Boston, MA: IEEE), 137–144.

Fidjeland, A., and Shanahan, M. (2010). "Accelerated simulation of spiking neural networks using GPUs," in *Neural Networks, International Joint Conference on* (Barcelona), 1–8.

Friedrich, J., Urbanczik, R., and Senn, W. (2011). Spatio-temporal credit assignment in neuronal population learning. *PLoS Computat. Biol.* 7:e1002092. doi: 10.1371/journal.pcbi.1002092

Furber, S., Galluppi, F., Temple, S., and Plana, A. (2014). The SpiNNaker project. *Proc. IEEE* 102, 652–665. doi: 10.1109/JPROC.2014.2304638

Furber, S., and Temple, S. (2008). Neural systems engineering. *Computat. Intell.* 4, 763–796. doi: 10.1098/rsif.2006.0177

Furber, S., Temple, S., and Brown, A. (2006). "High-performance computing for systems of spiking neurons," in *The AISB06 Workshop on GC5: Architecture of Brain and Mind* (Bristol).

Galluppi, F., Davies, S., Rast, A., Sharp, T., Plana, L., and Furber, S. (2012). "A hierarchical configuration system for a massively parallel neural hardware platform," in *CF '12 Proceedings of the 9th Conference on Computing Frontiers*, ed ACM (Cagliari), 183–192.

Galluppi, F., Denk, C., Meiner, M. C., Stewart, T., Plana, L. A., Eliasmith, C., et al. (2014). "Event-based neural computing on an autonomous mobile platform," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (Hong Kong) 2862–2867.

Gerstner, W., Kempter, R., van Hemmen, J. L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature* 383, 76–78. doi: 10.1038/383076a0

Gerstner, W., Sprekeler, H., and Deco, G. (2012). Theory and simulation in neuroscience. *Science* 338, 60–65. doi: 10.1126/science.1227356

Giulioni, M., Camilleri, P., Dante, V., Badoni, D., Indiveri, G., Braun, J., et al. (2008). "A VLSI network of spiking neurons with plastic fully configurable learning synapses," in *2008 15th IEEE International Conference on Electronics, Circuits and Systems* (St. Julien's).

Giulioni, M., Pannunzi, M., Badoni, D., Dante, V., and Del Giudice, P. (2009). Classification of correlated patterns with a configurable analog VLSI neural network of spiking neurons and self-regulating plastic synapses. *Neural Comput.* 21, 3106–3129. doi: 10.1162/neco.2009.08-07-599

Guyonneau, R., Van Rullen, R., and Thorpe, S. J. (2005). Neurons tune to the earliest spikes through STDP. *Neural Comput.* 17, 859–879. doi: 10.1162/0899766053429390

Hartline, H. K., Wagner, H. G., and Ratliff, F. (1956). Inhibition in the eye of the limulus. *J. Gen. Physiol.* 39, 651–673. doi: 10.1085/jgp.39.5.651

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. New York, NY: Wiley-Interscience.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. U.S.A.* 79, 2554–2558. doi: 10.1073/pnas.79.8.2554

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Indiveri, G., and Fusi, S. (2007). "Spike-based learning in VLSI networks of integrate-and-fire neurons," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on* (New Orleans, LA), 3371–3374.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., and Prodromakis, T. (2013). Integration of nanoscale memristor synapses in neuromorphic computing architectures. *IOP Nanotechnol.* 24:384010. doi: 10.1088/0957-4484/24/38/384010

Izhikevich, E. (2006). Polychronization: computation with spikes. *Neural Comput.* 18, 245–282. doi: 10.1162/089976606775093882

Izhikevich, E. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb. Cortex* 17, 2443–2452. doi: 10.1093/cercor/bhl152

Jeyabalaratnam, J., Bharmauria, V., Bachatene, L., Cattan, S., Angers, A., and Molotchnikoff, S. (2013). Adaptation shifts preferred orientation of tuning curve in the mouse visual cortex. *PLoS ONE* 8:e64294. doi: 10.1371/journal.pone.0064294

Jin, X., Galluppi, F., Patterson, C., Rast, A., Davies, S., Temple, S., et al. (2010a). "Algorithm and software for simulation of spiking neural networks on the multi-chip SpiNNaker system," in *Neural Networks, 2010. IJCNN 2010.*

(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on (Barcelona), 1–8.

Jin, X., Rast, A., Galluppi, F., Davies, S., and Furber, S. (2010b). "Implementing spike-timing-dependent plasticity on SpiNNaker neuromorphic hardware," in *Neural Networks, 2010. IJCNN 2010. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on* (Barcelona: IEEE), 1–8.

Jin, X., Rast, A., Galluppi, F., Khan, M., and Furber, S. (2009). "Implementing learning on the SpiNNaker universal neural chip multiprocessor," in *Neural Information Processing, Volume 5863, Chapter 48*, eds C. S.Leung, M. Lee, and J. H. Chan (Berlin; Heidelberg: Springer), 425–432.

Kappel, D., Nessler, B., and Maass, W. (2014). STDP installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLoS Computat. Biol.* 10:e1003511. doi: 10.1371/journal.pcbi.1003511

Kempter, R., Gerstner, W., and van Hemmen, J. L. (2001). Intrinsic stabilization of output rates by spike-based hebbian learning. *Neural Comput.* 13, 2709–2741. doi: 10.1162/089976601317098501

Kleberg, F. I., Fukai, T., and Gilson, M. (2014). Excitatory and inhibitory STDP jointly tune feedforward neural circuits to selectively propagate correlated spiking activity. *Front. Comput. Neurosci.* 8:53. doi: 10.3389/fncom.2014.00053

Kullmann, D. M., Moreau, A. W., Bakiri, Y., and Nicholson, E. (2012). Plasticity of inhibition. *Neuron* 75, 951–962. doi: 10.1016/j.neuron.2012.07.030

Lazar, A., Pipa, G., and Triesch, J. (2009). {SORN}: a self-organizing recurrent neural network. *Front. Comput. Neurosci.* 3:23. doi: 10.3389/neuro.10.023.2009

Le, Q. V., Monga, R., Devin, M., Corrado, G., Chen, K., Ranzato, M., et al. (2012). "Building high-level features using large scale unsupervised learning," in *Proceedings of the 29th International Conference on Machine Learning (ICML)* (Vancouver, BC), 1–11.

Legenstein, R., Pecevski, D., and Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Comput. Biol.* 4:e1000180. doi: 10.1371/journal.pcbi.1000180

Markram, H., Lbke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science* 275, 213–215. doi: 10.1126/science.275.5297.213

Mayr, C., Stärke, P., Partzsch, J., Schueffny, R., Cederstroem, L., Shuai, Y., et al. (2012). "Waveform driven plasticity in BiFeO3 memristive devices: model and implementation," in *NIPS* (Lake Tahoe, NV), 1709–1717.

Mead, C. (1989). *Analog VLSI and Neural Systems*. London: Addison-Wesley Longman Publishing Co., Inc.

Mehta, M. R., Quirk, M. C., and Wilson, M. A. (2000). Experience-dependent asymmetric shape of hippocampal receptive fields. *Neuron* 25, 707–715. doi: 10.1016/S0896-6273(00)81072-7

Min, R., Santello, M., and Nevian, T. (2012). The computational power of astrocyte mediated synaptic plasticity. *Front. Comput. Neurosci.* 6:93. doi: 10.3389/fncom.2012.00093

Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Circ. Syst.* 3, 32–42. doi: 10.1109/TBCAS.2008.2005781

Moore, B. D., and Freeman, R. D. (2012). Development of orientation tuning in simple cells of primary visual cortex. *J. Neurophysiol.* 107, 2506–2516. doi: 10.1152/jn.00719.2011

Morrison, A., Diesmann, M., and Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biol. Cybern.* 98, 459–478. doi: 10.1007/s00422-008-0233-1

Morrison, A., Mehring, C., and Geisel, T. (2005). Advancing the boundaries of high-connectivity network simulation with distributed computing. *Neural Comput.* 17, 1776–1801. doi: 10.1162/0899766054026648

Mostafa, H., Corradi, F., Stefanini, F., and Indiveri, G. (2014). "A hybrid analog/digital spike-timing dependent plasticity learning circuit for neuromorphic VLSI multi-neuron architectures," in *International Symposium on Circuits and Systems (ISCAS) 2014* (IEEE) 854–857.

Nageswaran, J., Dutt, N., Krichmar, J., and Nicolau, A. (2007). A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Netw.* 22, 791–800. doi: 10.1016/j.neunet.2009.06.028

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., and Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Front. Neurosci.* 7:272. doi: 10.3389/fnins.2013.00272

Neil, D., and Liu, S.-C. (2014). "Minitaur, an event-driven FPGA-based spiking network accelerator," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1–1.

Nessler, B., Pfeiffer, M., Buesing, L., and Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput. Biol.* 9:e1003037. doi: 10.1371/journal.pcbi.1003037

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *J. Math. Biol.* 15, 267–273. doi: 10.1007/BF00275687

Painkras, E., Plana, L., Garside, J., Temple, S., Galluppi, F., Patterson, C., et al. (2013). "SpiNNaker : a 1W 18-core system-on-chip for massively-parallel neural net simulation," in *The IEEE Journal of Solid State Circuits, VV, 8*, 1–13.

Pawlak, V., Wickens, J. R., Kirkwood, A., and Kerr, J. N. D. (2010). Timing is not everything: neuromodulation opens the STDP gate. *Front. Synaptic Neurosci.* 2:146. doi: 10.3389/fnsyn.2010.00146

Pfeil, T., Potjans, T. C., Schrader, S., Potjans, W., Schemmel, J., Diesmann, M., et al. (2012). Is a 4-Bit synaptic weight resolution enough? constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Front. Neurosci.* 6:90. doi: 10.3389/fnins.2012.00090

Plana, L., Furber, S., Temple, S., Khan, M., Shi, Y., Wu, J., et al. (2007). A GALS infrastructure for a massively parallel multiprocessor. *IEEE Des. Test Comput.* 24, 454–463. doi: 10.1109/MDT.2007.149

Plesser, H., Eppler, J., Morrison, A., Diesmann, M., and Gewaltig, M. (2007). "Efficient parallel simulation of large-scale neuronal networks on clusters of multiprocessor computers," in *Euro-Par 2007 Parallel Processing* (Berlin; Heidelberg: Springer), 672–681.

Potjans, W., Diesmann, M., and Morrison, A. (2011). An imperfect dopaminergic error signal can drive temporal-difference learning. *PLoS Comput. Biol.* 7:e1001133. doi: 10.1371/journal.pcbi.1001133

Rahimi Azghadi, M., Iannella, N., Al-Sarawi, S. F., Indiveri, G., and Abbott, D. (2014). Spike-based synaptic plasticity *in silicon*: design, implementation, application, and challenges. *Proc. IEEE* 102, 717–737. doi: 10.1109/JPROC.2014.2314454

Savin, C., Joshi, P., and Triesch, J. (2010). Independent component analysis in spiking neurons. *PLoS Comput. Biol.* 6:e1000757. doi: 10.1371/journal.pcbi.1000757

Schemmel, J., Bruderle, D., Meier, K., and Ostendorf, B. (2007). "Modeling synaptic plasticity within networks of highly accelerated I&F neurons," in *2007 IEEE International Symposium on Circuits and Systems* (New Orleans, LA). doi: 10.1109/ISCAS.2007.378289

Sharp, T., and Furber, S. (2013). "Correctness and performance of the SpiNNaker architecture," in *Neural Networks (IJCNN), The 2013 International Joint Conference on* (Dallas, TX), 1–8.

Sharp, T., Plana, L., Galluppi, F., and Furber, S. (2011). "Event-driven simulation of arbitrary spiking neural networks on SpiNNaker," in *The International Conference on Neural Information Processing (ICONIP), Volume 2011* (Shanghai: Springer), 424–430. doi: 10.1007/978-3-642-24965-5/48

Sheik, S., Coath, M., Indiveri, G., Denham, S. L., Wennekers, T., and Chicca, E. (2012). Emergent auditory feature tuning in a real-time neuromorphic VLSI system. *Front. Neurosci.* 6:17. doi: 10.3389/fnins.2012.00017

Shouval, H., Intrator, N., and Cooper, L. N. (1997). BCM network develops orientation selectivity and ocular dominance in natural scene environment. *Vis. Res.* 37, 3339–3342. doi: 10.1016/S0042-6989(97)00087-4

Sjöström, P. J., Turrigiano, G. G., and Nelson, S. B. (2001). Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron* 32, 1149–1164. doi: 10.1016/S0896-6273(01)00542-6

Song, S., and Abbott, L. F. (2001). Cortical development and remapping through spike timing-dependent plasticity. *Neuron* 32, 339–350. doi: 10.1016/S0896-6273(01)00451-2

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* 3, 919–926. doi: 10.1038/78829

Stewart, T. C., Tripp, B., and Eliasmith, C. (2009). Python scripting in the nengo simulator. *Front. Neuroinformat.* 3:7. doi: 10.3389/neuro.11.007.2009

Stromatias, E., Galluppi, F., Patterson, C., and Furber, S. (2013). "Power analysis of large-scale, real-time neural networks on SpiNNaker," in *The International Joint Conference on Neural Networks - ICJNN 2013* (Dallas, TX), 1570–1577.

Tapson, J. C., Cohen, G. K., Afshar, S., Stiefel, K. M., Buskila, Y., Wang, R. M., et al. (2013). Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Front. Neurosci.* 7:153. doi: 10.3389/fnins.2013.00153

van Schaik, A., and Liu, S. (2005). "AER EAR: a matched silicon cochlea pair with address event representation interface," in *IEEE International Symposium on Circuits and Systems ISCAS* (Kobe), 4213–4216.

Vogelstein, R. J., Tenore, F., Philipp, R., Adlerstein, M. S., Goldberg, D. H., and Cauwenberghs, G. (2002). "Spike timing-dependent plasticity in the address domain," in *Advances in Neural Information Processing Systems* (Vancouver, CA), 1147–1154.

Wang, R., Cohen, G., Stiefel, K. M., Hamilton, T. J., Tapson, J., and van Schaik, A. (2013). An FPGA implementation of a polychronous spiking neural network with delay adaptation. *Front. Neurosci.* 7:14. doi: 10.3389/fnins.2013.00014

Wong, T., Preissl, R., Datta, P., Flickner, M., Singh, R., Esser, S., et al. (2013). 10ˆ14. Technical report, IBM.

Zamarreño Ramos, C., Camuñas Mesa, L. A., Pérez-Carrasco, J. A., Masquelier, T., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex. *Front. Neurosci* 5:26. doi: 10.3389/fnins.2011.00026

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 11 July 2014; accepted: 07 December 2014; published online: 20 January 2015.

Citation: Galluppi F, Lagorce X, Stromatias E, Pfeiffer M, Plana LA, Furber SB and Benosman RB (2015) A framework for plasticity implementation on the SpiNNaker neural architecture. Front. Neurosci. ***8****:429. doi: 10.3389/fnins.2014.00429*

This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.

APPENDIX: PSEUDO CODE

This appendix presents pseudo code describing the implementation of the common framework and of the rules proposed in this paper. We refer to DTCM for the memory locally available to each ARM core, and to SDRAM as the off-die memory shared by all cores in a chip.

A COMMON CODE

The infrastructure for the framework is common for all the learning rules introduced, leaving users free to implement their plasticity rules starting from:

- spike windows containing a compressed form of spike timings for the post-synaptic neurons (stored in DTCM).
- the whole synaptic matrix, indexed by pre-synaptic neuron (stored in SDRAM). Each row contains all the synaptic information needed by the post-synaptic neurons modeled by the neural core for a particular pre-synaptic neuron.
- spike windows containing a compressed form of spike timings for the pre-synaptic neurons (stored at the beginning of each pre-synaptic row in the synaptic matrix).
- additional data needed can be passed through means of shared portions of SDRAM and initialized during the timer callback.

The DMA controller is managing access to SDRAM, relieving the ARM cores of this task and enabling them to perform computation while the memory is fetched or written in parallel. When a core needs to access SDRAM it issues a request to the DMA controller and continues its work. Whenever the requested access is completed a *DMA done interrupt* is generated, triggering a *DMA done callback*.

Synaptic data is indexed by pre-synaptic neuron in SDRAM; each row contains the information about all the core-local post-synaptic neurons each pre-neuron is connected to. Each iteration of the DMA pipeline calls itself on the next portion of the synaptic matrix if there is still data to compute.

Algorithm 1 | DMA Pipeline.

```
Data: Local copy of synaptic row data: synaptic_row
Result: Starts the weight update process; requests the next synaptic row
if there is still synaptic data to process then
    Increment the synaptic_matrix_index;
    Issue a DMA request for synaptic row at address synaptic_matrix_index;
    Call the weight update process on synaptic_row;
else
    Terminate the DMA pipeline process;
end
Cleanup local variables for the next plasticity phase;
```

The *Packet Received* callback is called whenever a MC packet (containing a post spike) is received, and is used to updated the post-spike window. The mapping software ensures that every spike emitted by the neural core reaches the correspondent plasticity core.

Algorithm 2 | Packet Received callback.

```
Data: Incoming spike (routing key) from the twin neural core
Result: Spike window for the current phase
Retrieve the spike window for the post-synaptic neuron (DTCM);
Retrieve the time position for the spike;
Update the spike window;
```

The *Timer callback* is called every millisecond, and initiates the plasticity process every 128 ms in our implementation. The maximum axonal delay is reintroduced here, to ensure that all the necessary information from the previous phase is present. When initiating the plasticity process rule-specific variables (e.g., traces) can be computed; therefore the timer callback is dependent of each plasticity rule, and described in the next Sections for the three rules implemented.

The *weight update* process is initiated whenever the next row from the synaptic matrix has been retrieved from SDRAM and is locally available to a core. Similarly to the *timer callback*, this function is specific to each plasticity rule implemented and is therefore described in the following Sections.

B STDP

Algorithm 3 | Timer callback.

```
Data: Time tick
Compute phase index from time tick;
if mod(time tick,plasticity_update_time) == max_delay then
    Compute post-synaptic traces from the post-spike window of the previous phase (DTCM);
    Start DMA pipeline process;
else
    do nothing;
end
```

Algorithm 4 | STDP weight update process.

```
Data: local copy of a synaptic row for pre-neuron $i$
Result: updated synaptic row
Compute the pre-synaptic trace from the pre-spike window bitmap (SDRAM);
/* potentiation                                                    */
for $j$ in size_row do
    while there are post-spikes for neuron $j$ do
        Increment $\Delta w_{ij}$ using the pre_trace value at time $t^j_{post}$
    end
end
/* depotentiation                                                  */
while there are pre-spikes for neuron $i$ do
    for $j$ in size_row do
        Decrement $\Delta w_{ij}$ using the post_trace value at time $t^i_{pre}$
    end
end
for $j$ in size_row do
    Apply $\Delta w_{ij}$ to the synapse;
end
Cleanup spike window;
Initiate a DMA write-back of the synaptic row (DTCM $\rightarrow$ SDRAM)
```

C BCM

Algorithm 5 | Timer callback.

```
Data: Time tick
Compute phase index from time tick;
if mod(time tick,plasticity_update_time) == max_delay then
    Compute post-synaptic rates from the spike window of the previous phase (DTCM);
    Start DMA pipeline process;
else
    do nothing;
end
```

Algorithm 6 | BCM weight update process.

```
Data: local copy of a synaptic row for pre-neuron $i$
Result: updated synaptic row
Compute the pre-synaptic rate from the pre-spike window;
for $j$ in size_row do
    Compute $\Delta w_{ij}$ using Eq. 4;
end
Cleanup spike window;
Initiate a DMA write-back of the synaptic row (DTCM $\rightarrow$ SDRAM);
```

D VOLTAGE-GATED STDP

Algorithm 7 | Timer callback.

```
Data: Time tick
Compute phase index from time tick;
if mod(time tick,plasticity_update_time) == max_delay then
    Compute $C(t)$ traces from the spike window of the previous phase (DTCM);
    Retrieve voltage threshold checks from a shared portion of SDRAM;
    Start DMA pipeline process;
else
    do nothing;
end
```

Algorithm 8 | Voltage-gated STDP weight update process.

```
Data: local copy of a synaptic row for pre-neuron $i$
Result: updated synaptic row
while there are pre-spikes for neuron $i$ do
    for $j$ in size_row do
        if $V(t^i_{pre})$ and $C(t^i_{pre})$ satisfy Eq. 6 then
            increment $\Delta w_{ij} \leftarrow +a$;
        else if $V(t^i_{pre})$ and $C(t^i_{pre})$ satisfy Eq. 7 then
            decrement $\Delta w_{ij} \leftarrow -b$;
        else
            do nothing;
        /* relax to one of the bistable states */
        if $w$ satisfies Eq. 8 then
            relax toward $w_{max}$;
        else
            relax toward $w_{min}$;
        end
    end
end
Cleanup spike window;
Initiate a DMA write-back of the synaptic row (DTCM $\rightarrow$ SDRAM);
```

ORIGINAL RESEARCH ARTICLE
published: 22 January 2015
doi: 10.3389/fnins.2015.00002

Reducing the computational footprint for real-time BCPNN learning

Bernhard Vogginger[1]*, René Schüffny[1], Anders Lansner[2,3], Love Cederström[1], Johannes Partzsch[1] and Sebastian Höppner[1]

[1] Department of Electrical Engineering and Information Technology, Technische Universität Dresden, Germany
[2] Department of Computational Biology, School of Computer Science and Communication, Royal Institute of Technology (KTH), Stockholm, Sweden
[3] Department of Numerical Analysis and Computer Science, Stockholm University, Stockholm, Sweden

Edited by:
Sadique Sheik, University of Zurich and ETH Zurich, Switzerland

Reviewed by:
Guillaume Garreau, Johns Hopkins University, USA
Sergio Davies, The University of Manchester, UK
Srinjoy Das, University of California, San Diego, USA

***Correspondence:**
Bernhard Vogginger, Chair for Highly Parallel VLSI Systems and Neuromorphic Circuits, Institue of Circuits and Systems, Department of Electrical Engineering and Information Technology, Technische Universität Dresden, 01062 Dresden, Germany
e-mail: bernhard.vogginger@tu-dresden.de

The implementation of synaptic plasticity in neural simulation or neuromorphic hardware is usually very resource-intensive, often requiring a compromise between efficiency and flexibility. A versatile, but computationally-expensive plasticity mechanism is provided by the Bayesian Confidence Propagation Neural Network (BCPNN) paradigm. Building upon Bayesian statistics, and having clear links to biological plasticity processes, the BCPNN learning rule has been applied in many fields, ranging from data classification, associative memory, reward-based learning, probabilistic inference to cortical attractor memory networks. In the spike-based version of this learning rule the pre-, postsynaptic and coincident activity is traced in three low-pass-filtering stages, requiring a total of eight state variables, whose dynamics are typically simulated with the fixed step size Euler method. We derive analytic solutions allowing an efficient event-driven implementation of this learning rule. Further speedup is achieved by first rewriting the model which reduces the number of basic arithmetic operations per update to one half, and second by using look-up tables for the frequently calculated exponential decay. Ultimately, in a typical use case, the simulation using our approach is more than one order of magnitude faster than with the fixed step size Euler method. Aiming for a small memory footprint per BCPNN synapse, we also evaluate the use of fixed-point numbers for the state variables, and assess the number of bits required to achieve same or better accuracy than with the conventional explicit Euler method. All of this will allow a real-time simulation of a reduced cortex model based on BCPNN in high performance computing. More important, with the analytic solution at hand and due to the reduced memory bandwidth, the learning rule can be efficiently implemented in dedicated or existing digital neuromorphic hardware.

Keywords: Bayesian confidence propagation neural network (BCPNN), Hebbian learning, synaptic plasticity, event-driven simulation, spiking neural networks, look-up tables, fixed-point accuracy, digital neuromorphic hardware

1. INTRODUCTION

Bayesian Confidence Propagation Neural Networks (BCPNNs) realize Bayesian statistics with spiking or non-spiking neural networks. They can be used to build powerful associative memories (Sandberg et al., 2000; Meli and Lansner, 2013) and data classifiers, with applications ranging from data mining (Bate et al., 1998; Lindquist et al., 2000) to olfaction modeling (Kaplan and Lansner, 2014). The underlying Bayesian learning rule has clear links to biological synaptic plasticity processes (Tully et al., 2014), cortical associative memory (Lansner, 2009), reinforcement learning (Johansson et al., 2003), and action selection (Berthet et al., 2012). Furthermore, BCPNNs have been used to model phenomena like synaptic working memory (Sandberg et al., 2003), word-list learning in humans (Lansner et al., 2013) and memory consolidation (Fiebig and Lansner, 2014), making it a promising paradigm for information processing in the brain, while retaining a level of abstraction suitable for efficient technical implementation. Models using more detailed spiking attractor networks with the same structure have provided non-trivial explanations for memory retrieval and other basic cognitive phenomena like e.g., attentional blink (Lundqvist et al., 2010, 2011; Silverstein and Lansner, 2011; Lundqvist et al., 2013).

The performance of BCPNNs, for example in memory tasks, scales well with network size, making them extraordinarily powerful for large networks (Johansson et al., 2001). Therefore, massively parallel simulations of these networks (29 million spiking units, 295 billion plastic connections) have been realized on supercomputers (Benjaminsson and Lansner, 2011). These showed that BCPNN implementations are bounded by computation (Johansson and Lansner, 2007). To alleviate this limit, conceptual work on implementations in neuromorphic hardware has been performed (Johansson and Lansner, 2004; Farahini et al., 2014; Lansner et al., 2014).

In this paper, we pave the way for an efficient implementation of BCPNN in digital neuromorphic hardware by reducing both its computational and memory footprint. Existing software models apply fixed step size numerical integration methods for solving the BCPNN dynamics. Although easy to implement, this clock-driven simulation approach has two major drawbacks: First, there is a relatively high base cost for calculating the updates of all state variables at every time step, irrespective of the spiking activity in the network. Second, the states have to be read from and written back to memory at every simulation step, which is especially expensive for custom hardware implementations where the states are stored in an external memory. As suggested in recent work (Lansner et al., 2014), we tackle these issues by moving to an event-driven simulation scheme, which we systematically optimize for minimal number of calculations to achieve a reduction of the computational load by an order of magnitude. This efficiency gain of the event-driven paradigm is mainly due to the sparse activity in BCPNNs, which is retained irrespective of network size. Employing pre-calculated look-up tables for the frequent calculation of the exponential function, we further minimize the computational cost per event-driven update. By using an analytical solution of the model equations, the numerical accuracy of the simulation is increased compared to conventional simulation techniques with fixed step size (Henker et al., 2012). We show how this accuracy overhead could be utilized for significantly reducing the required memory and memory bandwidth in a potential hardware implementation by using fixed point operands with fewer bits than in a floating point representation.

While we performed our equation optimizations specifically for the BCPNN model, they are not restricted to it. As BCPNNs rely on dynamic equations that are common in neuroscientific modeling, our approach can be easily adopted to other models. It shows how to efficiently calculate single neuronal traces and correlation measures for synaptic plasticity, increasing the energy efficiency of digital implementations, either on standard computers or on specialized hardware, on an algorithmic level, complementing analog approaches for increasing the energy efficiency of neuromorphic computation (Hasler and Marr, 2013).

2. MATERIALS AND METHODS

2.1. BAYESIAN CONFIDENCE PROPAGATION NEURAL NETWORKS

In BCPNNs (Lansner and Ekeberg, 1989; Lansner and Holst, 1996) the synaptic weights between network units are calculated in a Hebbian fashion by applying Bayes' rule on the past activity of the units giving a measure of the co-activation of the units. In a similar manner each unit's bias is calculated from its past activity, representing its *a priori* probability to be active. Often, the activity of the units is represented by stochastic spike events, which are generated according to each unit's recent input and own activity. Typically, in a *training* phase these correlation and activation statistics are collected, which are then used in the subsequent *test* phase to perform inference, i.e., to determine the *a posteriori* activity of some units as a response to other units' recent activity. While the concept of BCPNN was originally developed for series of discrete samples, a time-continuous spike-based version has been developed recently, which we describe in Section 2.1.1 and whose efficient simulation is the main subject of this article. In Section 2.1.2, we present an application of this spike-based BCPNN learning rule in a modular network that constitutes a reduced full-scale model of the cortex.

2.1.1. Spike-based BCPNN

Spike-based BCPNN (Wahlgren and Lansner, 2001; Tully et al., 2014) is implemented by a set of local synaptic state variables that keep track of presynaptic, postsynaptic, and synaptic (i.e., correlated) activity over three different time scales, by passing spiking activity over three low pass filters, see **Figure 1**. Here and throughout this paper the three sites (pre-, postsynaptic and synaptic) are denoted by indices i, j, and ij, respectively. In the first processing stage, the pre- and postsynaptic spiking activity represented by spike trains S_i (resp. S_j) is low pass filtered into the Z_i and Z_j traces (**Figure 1B**), with time constants τ_{z_i} and τ_{z_j} in a range of 5 ms to 100 ms, which corresponds to typical synaptic decay time constants for various receptor types.

In the second stage, the Z traces are passed on to the E or eligibility traces and low pass filtered with time constant τ_e. Here, a separate trace E_{ij} is introduced to filter the coincident activity of the Z-traces, see **Figure 1C**. The E traces typically have slower dynamics than the Z traces ($\tau_e \approx 20 - 1000$ ms), and can be motivated to provide a mechanism for delayed reward learning (cf. Tully et al., 2014).

The E traces in turn are low pass filtered into the P traces (**Figure 1D**). These tertiary traces have the slowest dynamics with time constant τ_p ranging from 1 s to several 100 s, even higher values are possible. The P traces correspond to the probabilities of the units being active or co-active in the original non-spiking BCPNN formulation (Lansner and Holst, 1996). In a final step the P traces are used to compute the synaptic weight w_{ij} and the postsynaptic bias β_j (**Figure 1E**). The formulas for w_{ij} and β_j contain the parameter ϵ, which originates from a minimum spiking activity assumed for the pre- and postsynaptic units (cf. Tully et al., 2014), and which has the side effect to avoid division by zero in the weight formula.

The global parameter κ in the dynamics of P traces can take any non-negative value and controls the learning, i.e., it determines how strong recent correlations are stored. When the learning rate κ equals zero, there is no learning, as the P traces do not change at all, and thus neither do the synaptic weight w_{ij} and the postsynaptic bias β_j. We assume that κ only undergoes discrete and seldom changes, mostly when learning is switched on or off. Hence, while κ is constant and non-zero, the dynamics of the P traces can be expressed with a modified time constant τ_p^*:

$$\tau_p^* \frac{dP}{dt} = E - P, \qquad \tau_p^* = \frac{\tau_p}{\kappa} \tag{1}$$

We refer to Tully et al. (2014) for establishing the link between the spike-based and the probabilistic BCPNN learning rule, as well as for details on the biological equivalents of the processing stages. Also, note that in some cases the second low pass filter is not actually used, so that the Z traces are directly passed to the P traces.

2.1.2. Reduced modular model of the cortex

As an application of the spike-based BCPNN we consider a modular abstract network model, motivated by the columnar structure of the cortex, that was already presented in Lansner et al. (2014). One assumption is that the smallest functional units

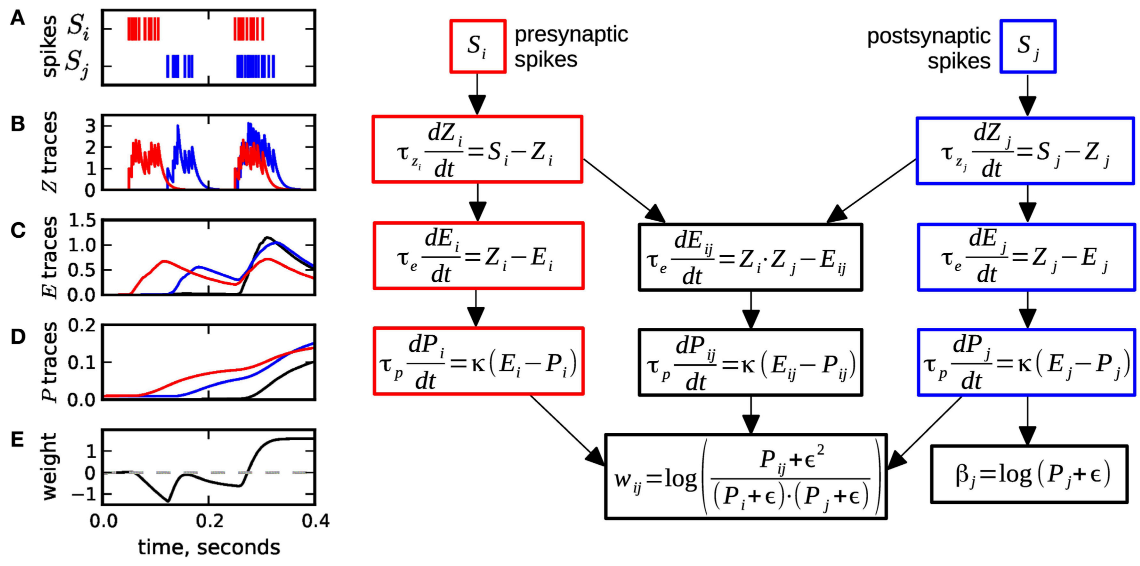

A
spikes
S_i
S_j
B
Z traces
C
E traces
D
P traces
E
weight
time, seconds
presynaptic spikes
postsynaptic spikes
$\tau_{z_i}\frac{dZ_i}{dt}=S_i-Z_i$
$\tau_{z_j}\frac{dZ_j}{dt}=S_j-Z_j$
$\tau_e\frac{dE_i}{dt}=Z_i-E_i$
$\tau_e\frac{dE_{ij}}{dt}=Z_i\cdot Z_j-E_{ij}$
$\tau_e\frac{dE_j}{dt}=Z_j-E_j$
$\tau_p\frac{dP_i}{dt}=\kappa(E_i-P_i)$
$\tau_p\frac{dP_{ij}}{dt}=\kappa(E_{ij}-P_{ij})$
$\tau_p\frac{dP_j}{dt}=\kappa(E_j-P_j)$
$w_{ij}=\log\left(\frac{P_{ij}+\epsilon^2}{(P_i+\epsilon)\cdot(P_j+\epsilon)}\right)$
$\beta_j=\log(P_j+\epsilon)$

FIGURE 1 | Equations and sample traces of the spike-based BCPNN learning rule. (A) Presynaptic (red) S_i and postsynaptic (blue) S_j spike trains serve as input to a BCPNN synapse. **(B)** The input spike trains are low pass filtered into the Z traces with time constants τ_{z_i}, τ_{z_j}. **(C)** E traces compute the low-pass filter of Z traces with τ_e. The E_{ij} variable (black) tracks coincident pre- and postsynaptic activity. **(D)** E traces are passed on to the P traces and low-pass filtered with τ_p. **(E)** The P traces are used to compute the postsynaptic bias β_j and the synaptic weight w_{ij}, which can vary between positive and negative values. Usually, the dynamics get slower from **B** to **D:** $\tau_{z_i}, \tau_{z_j} \le \tau_e \le \tau_p$. Figure redrawn from Tully et al. (2014).

in the mammalian cortex are not single neurons but so-called minicolumns. A minicolumn is formed by a local population of some hundred neurons with enhanced recurrent connectivity and similar receptive fields, so that these neurons are assumed to have quite correlated output. An example would be a minicolumn encoding a certain orientation during processing in primary visual cortex.

In the order of 100 minicolumns are aggregated in a larger columnar structure, the cortical hypercolumn, which contains in the order of 10,000 neurons. Within a hypercolumn the minincolumns compete in a soft winner-take all (soft-WTA) fashion through feedback inhibition, so that most of the time only one minicolumn shows high firing activity while the others are mostly silent. Minicolumns can be viewed to encode a discrete value of an attribute specific to each hypercolumn.

In our reduced, abstract model, each minicolumn is represented by one stochastically spiking minicolumn unit (MCU). Only connections outside a hypercolumn are implemented: The internal connections between neurons of a minicolumn are hidden within the MCU, while the competitive feedback inhibition of the 100 MCUs within a hypercolumn unit (HCU) is hardwired by means of a normalization of activity per HCU (cf. Equation 5 below). In turn, for the implementation of the incoming long-range synaptic connections, which on the neuron level typically make up half of the between 10^3 and 10^4 incoming connections in total, we assume that each MCU propagates its spikes to 10,000 other MCUs, and has appropriately as many incoming connections. These connections are *patchy* in the sense that each MCU projects onto 100 hypercolumns and delivers spikes to all 100 MCUs of each target HCU. The connection scheme is motivated as follows: Long-range connections are provided by large layer 5 pyramidal cells, which make up around 10% of a minicolumn. Each of those cells forms synaptic connections to clusters of far away neurons in horizontal direction. The diameter of these clusters approximately corresponds to the dimension of a hypercolumn. In real cortex, each of the large pyramidal cells generates around 10 of these *patches* (Houzel et al., 1994; Binzegger et al., 2007), which motivates the 100 target HCUs per MCU, assuming that one MCU comprises one hundred neurons. All of these connections between MCUs are subject to the spike-based BCPNN learning equations of **Figure 1**.

At a higher level, HCUs represent independent network modules between which spikes are transmitted. Each HCU consists of 100 MCUs and 1 million plastic synapses organized in an array with 10^4 inputs and 100 outputs, as illustrated in **Figure 2**. The pre- and postsynaptic states of the BCPNN model can therefore be implemented at the margin of the array, while the synaptic traces E_{ij},P_{ij}, and w_{ij} form the array, thus representing the largest amount of state variables. The minicolumn units integrate the incoming spiking activity, which is then turned into a spiking probability of each unit. In particular, presynaptic input leads to a synaptic current $s_{\text{syn},\text{j}}$ (Equation 2), which together with the bias β_j and a specific external input I_j sums up to the support value s_j for each minicolum unit j in Equation (3):

$$\tau_{z_i}\frac{ds_{\text{syn},\text{j}}(t)}{dt} = \sum_i w_{ij}(t)S_i(t) - s_{\text{syn},\text{j}}(t) \tag{2}$$

$$s_j(t) = \beta_j(t) + s_{\text{syn},\text{j}}(t) + I_j(t) \quad . \tag{3}$$

The low-pass filtered version of Equation (3) gives the "membrane potential" m_j of each MCU:

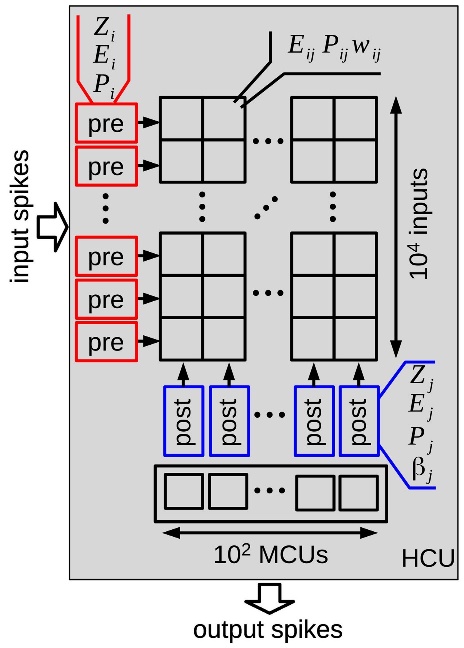


FIGURE 2 | Structure of one hypercolumn unit (HCU) of the reduced cortex model. Each HCU contains 100 stochastic minicolumn units (MCUs) that compete in a winner-take-all fashion. Input spikes from 10,000 MCUs of other HCUs connect via 1 million BCPNN synapses to all 100 MCUs of the HCU. Each MCU sends its output spikes to 100 other HCUs. In order to store all *Z*,*E*,*P* traces and synaptic weights of the HCU, more than 12 MB memory is required when using floating point numbers with single-precision.

$$\tau_m \frac{dm_j(t)}{dt} = s_j(t) - m_j(t) \quad , \tag{4}$$

where τ_m is the membrane time constant in the order of 10 ms. In other words, the MCUs are leaky-integrators (Equation 4) with three different input currents (Equation 3): The bias $\beta_j(t)$ represents the *prior* contribution to the unit's activation irrespective of the current synaptic input, determined by the past spiking activity of the unit itself via the postsynaptic traces (Z_j, E_j, P_j, cf. **Figure 1**). The synaptic input is implemented as an exponentially decaying synaptic current $s_{\text{syn},j}(t)$ (Equation 2), which - at a presynaptic spike of input i - is increased by synaptic weight $w_{ij}(t)$ learned according to the Equations in **Figure 1**. Last, the external input $I_j(t)$ allows a specific stimulation of single units.

All M MCUs of a hypercolumn unit are organized as a probabilistic soft-WTA circuit. The activation o_j of each unit is computed as:

$$o_j = \begin{cases} \dfrac{e^{\gamma_m m_j}}{\sum_{k=1}^{M} e^{\gamma_m m_k}}, & \text{if} \sum_{k=1}^{m} e^{\gamma_m m_k} > 1 \\ e^{\gamma_m m_j} & \text{otherwise} \end{cases} \quad , \tag{5}$$

The gain factor γ_m controls the strength of the soft-WTA filtering process, the higher γ_m the higher the activation-ratio between the winning unit and the remaining units. The normalization in Equation (5) ensures that on average not more than 1 MCU is active at the same time.

The activation o_j then translates into the instantaneous Poisson firing rate r_j for each unit:

$$r_j(t) = o_j(t) \cdot r_{\text{max,HCU}} \tag{6}$$

where $r_{\text{max,HCU}}$ is the maximum firing rate per HCU. The average spiking frequency in mammalian cortex is quite sparse, with an average spike rate on the order of 0.1 Hz (Lennie, 2003). In our full scale HCU with 100 MCUs the average activity level would be around 1 Hz (thus $r_{\text{max,HCU}} = 100$ HZ), and the difference is explained by the fact that one MCU represents around 10 layer 5 pyramidal cells.

2.2. SIMULATION STRATEGIES

2.2.1. *Fixed step size simulation*

The typical approach for the simulation of spiking neural networks is simulation with fixed step size, where all states are synchronously updated at every tick of a clock (Brette et al., 2007; Henker et al., 2012). Usually, in such time-driven simulation, one uses numerical integration methods like Euler or Runge-Kutta to advance the state by one time step *dt*.

For our reference fixed step size simulation we follow Lansner et al. (2014) and use the explicit Euler method for the numerical integration with a rather long time step of $dt = 1$ ms. As the MCUs are stochastic, the instantaneous firing rate r_j (Equation 6) is transformed into a firing probability per time step, which is then compared to a uniform random number between 0 and 1 to generate spikes. The 1 ms time step is also used in state-of-the-art real-time digital neuromorphic systems like the SpiNNaker (Furber et al., 2014) and the Synapse hardware (Merolla et al., 2014). For completeness, we also present results with 0.1 ms step size, which is commonly used for the simulation of spiking neural networks.

2.2.2. *Event-driven simulation*

In Sections 2.3.1 and 2.3.3 we provide analytical solutions for the spike-based BCPNN model. For those simulations we mix the time-driven and event-driven approach: We restrict spike times to multiples of the simulation time step *dt*. The stochastic MCUs (Equations 2–6) are evaluated as for the time-driven approach, which requires that also the β_j is computed at every time step. In contrast, the states of the BCPNN synapses (**Figure 1**) are only updated at the occurrence of a pre- or postsynaptic event.

2.3. ANALYTICAL SOLUTIONS OF SPIKE-BASED BCPNN

The simulation of spike-based BCPNN with a fixed step size method is cost-intensive and requires very frequent read and write of the state variables from and to memory. Therefore, we first provide the rather straightforward analytical solution of the BCPNN equations in Section 2.3.1, allowing an exact event-driven simulation scheme. As intermediate step, we rewrite the BCPNN dynamics as a spike response model in Section 2.3.2, which then provides the basis for a second analytical solution with reduced number of operations (Section 2.3.3). Although not employed in the experiments in this article, discrete changes of the learning rate κ must be considered for the completeness of the two analytical solutions, which is done in Appendix A3 .

2.3.1. *BCPNN solution: analytical I*

For the event-driven simulation of BCPNN, the update of the state variables is only triggered by events (usually pre- or postsynaptic spikes). For each state variable one requires the time of its last update t^{last}, in contrast to the time-driven simulation, where all states correspond to the same global time. Event-driven simulations are especially efficient if the update of the states from t^{last} to the current time t can be solved analytically. Without further derivation, we give the analytic solution to advance the Z, E and P traces by $\Delta t = t - t^{\text{last}}$ from time t^{last} to t, provided that there is no spike between t^{last} and t. For the presynaptic traces the solutions are

$$Z_i(t) = Z_i(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_{z_i}}} + S_i(t) \tag{7}$$

$$E_i(t) = E_i(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_e}} + Z_i(t^{\text{last}}) a_i \left(\mathrm{e}^{-\frac{\Delta t}{\tau_{z_i}}} - \mathrm{e}^{-\frac{\Delta t}{\tau_e}}\right) \tag{8}$$

$$P_i(t) = P_i(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}} + a_i b_i \left(\mathrm{e}^{-\frac{\Delta t}{\tau_{z_i}}} - \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}}\right) Z_i(t^{\text{last}}) + \left(E_i(t^{\text{last}}) - a_i Z_i(t^{\text{last}})\right) c \left(\mathrm{e}^{-\frac{\Delta t}{\tau_e}} - \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}}\right) \quad , \tag{9}$$

with the following coefficients used for brevity:

$$a_i = \frac{\tau_{z_i}}{\tau_{z_i} - \tau_e} \quad , \quad b_i = \frac{\tau_{z_i}}{\tau_{z_i} - \tau_p^*} \quad , \quad c = \frac{\tau_e}{\tau_e - \tau_p^*} \quad . \tag{10}$$

In Equation (7) S_i describes the presynaptic spike train taking value 1 at the spike time t_i^f and value 0 otherwise, formally

$$S_i(t) = \textstyle\sum_{t_i^f} \delta(t - t_i^f) \quad , \tag{11}$$

where $\delta(\cdot)$ denotes a Dirac pulse. We note that Equation (8) is only valid when $\tau_{z_i} \neq \tau_e$, Equation (9) furthermore requires that τ_p^* is different from both τ_{z_i} and τ_e. For the sake of simplicity we restrict ourselves within this article to time constants fulfilling this condition, but give the solution for the other cases in Appendix A.

The update formulas for the postsynaptic traces Z_j, E_j, and P_j can be obtained by replacing indices i by j in the presynaptic update formulas.

Accordingly, the update of the synaptic traces E_{ij} and P_{ij} is given by:

$$E_{ij}(t) = E_{ij}(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_e}} + Z_i(t^{\text{last}}) Z_j(t^{\text{last}}) a_{ij} \left(\mathrm{e}^{-\frac{\Delta t}{\tau_{z_{ij}}}} - \mathrm{e}^{-\frac{\Delta t}{\tau_e}}\right) \tag{12}$$

$$P_{ij}(t) = P_{ij}(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}} + a_{ij} b_{ij} \left(\mathrm{e}^{-\frac{\Delta t}{\tau_{z_{ij}}}} - \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}}\right) Z_i(t^{\text{last}}) Z_j(t^{\text{last}}) + \left(E_{ij}(t^{\text{last}}) - a_{ij} Z_i(t^{\text{last}}) Z_j(t^{\text{last}})\right) c \left(\mathrm{e}^{-\frac{\Delta t}{\tau_e}} - \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}}\right), \tag{13}$$

with shortcuts

$$\tau_{z_{ij}} = \left(\frac{1}{\tau_{z_i}} + \frac{1}{\tau_{z_j}}\right)^{-1} \quad , a_{ij} = \frac{\tau_{z_{ij}}}{\tau_{z_{ij}} - \tau_e} \quad , b_{ij} = \frac{\tau_{z_{ij}}}{\tau_{z_{ij}} - \tau_p^*} \quad . \tag{14}$$

Note that, on purpose, Equations (9, 13) were not further simplified to ease the comparison with the spike response model formulation of the BCPNN model in the next section. Again, we restrict ourselves to parameter sets where none of the involved time constants ($\tau_{z_{ij}}$, τ_e and τ_p^*) are equal. Note, however, that τ_{z_i} and τ_{z_j} may be equal.

The analytical solution of the BCPNN equations derived in this section is henceforth denoted as *analytical I* method.

2.3.2. *Spike response model formulation of the BCPNN model*

As starting point for a second event-driven analytical solution with less operations, we make use of the linearity of the BCPNN differential equations and formulate the dynamics as a spike response model, in accordance with the work of Gerstner and Kistler (2002). The presynaptic traces can be written as a response to spike times t_i^f:

$$Z_i(t) = \sum_{t_i^f} \zeta_i(t - t_i^f), \quad \zeta_i(t) = \mathrm{e}^{-\frac{t}{\tau_{z_i}}} \Theta(t) \tag{15}$$

$$E_i(t) = \sum_{t_i^f} \alpha_i(t - t_i^f), \quad \alpha_i(t) = a_i \left(\mathrm{e}^{-\frac{t}{\tau_{z_i}}} - \mathrm{e}^{-\frac{t}{\tau_e}}\right) \Theta(t) \tag{16}$$

$$P_i(t) = \sum_{t_i^f} \pi_i(t - t_i^f), \quad \pi_i(t) = a_i \left[b_i \left(\mathrm{e}^{-\frac{t}{\tau_{z_i}}} - \mathrm{e}^{-\frac{t}{\tau_p^*}}\right) + c \left(\mathrm{e}^{-\frac{t}{\tau_p^*}} - \mathrm{e}^{-\frac{t}{\tau_e}}\right)\right] \Theta(t). \tag{17}$$

Here $\Theta(\cdot)$ denotes the Heaviside step function. ζ_i, α_i, and π_i are the spike response kernels for the Z_i, E_i and P_i traces. One can obtain Equations (15–17) from the analytical solution by setting $Z_i(t^{\text{last}}) = 1, E_i(t^{\text{last}}) = 0, P_i(t^{\text{last}}) = 0$ in Equations (7–9). The spike response kernels ζ_i, α_i, and π_i are shown in the left panel of **Figure 3** as dashed lines. The postsynaptic traces can be analogously formulated, by replacing i with j in Equations (15–17).

For the synaptic trace variables E_{ij} and P_{ij} the spike response formulation becomes more sophisticated: Therefore, we consider the product $Z_i Z_j$, which after inserting the spike response formulation of Z_i and Z_j is given by:

$$Z_i Z_j = \sum_{t_i^f} \zeta_i(t - t_i^f) \cdot \sum_{t_j^f} \zeta_j(t - t_j^f) \tag{18}$$

$$= \sum_{t_i^f} \sum_{t_j^f} \mathrm{e}^{-\frac{t - t_i^f}{\tau_{z_i}}} \mathrm{e}^{-\frac{t - t_j^f}{\tau_{z_j}}} \Theta(t - t_i^f) \Theta(t - t_j^f) \tag{19}$$

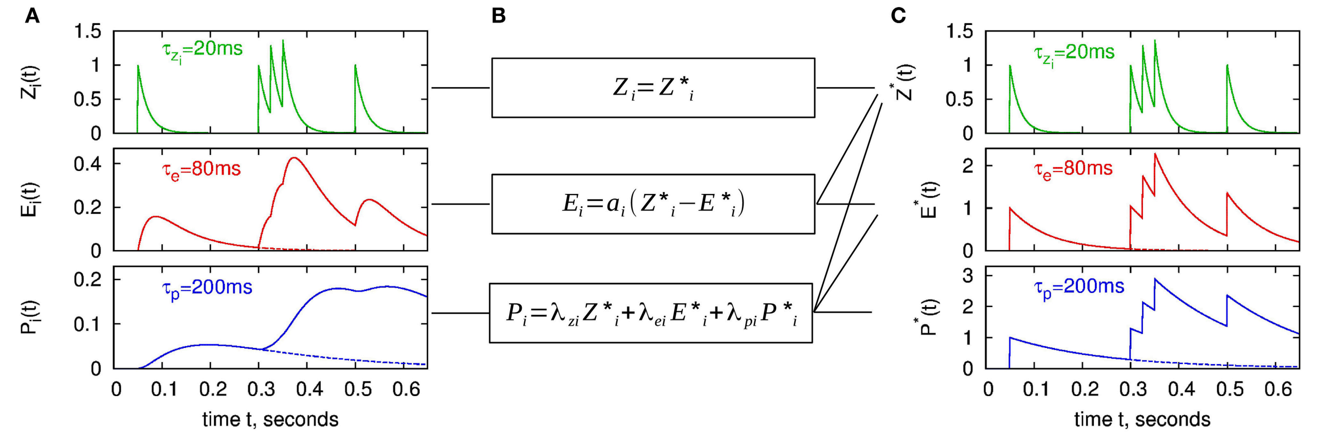


FIGURE 3 | BCPNN dynamics in two different representations for an example spike train. (A) Presynaptic traces according to the original model formulation (Z_i, E_i, P_i), as expressed in Equations (7–9). **(C)** "Exponential" traces Z_i^*, E_i^*, P_i^* used for the analytical II solution with minimal number of calculations, according to Equations (32–34). Dashed lines in **A** and **C** denote spike response kernels, i.e., responses to a single spike. **(B)** Transformation between the two different representations (Equations 29, 30, 46).

$$= \sum_{t_i^f} Z_j(t_i^f) e^{-(t-t_i^f)(\frac{1}{\tau_{z_i}} + \frac{1}{\tau_{z_j}})} \Theta(t - t_i^f)$$

$$+ \sum_{t_j^f} Z_i(t_j^f) e^{-(t-t_j^f)(\frac{1}{\tau_{z_i}} + \frac{1}{\tau_{z_j}})} \Theta(t - t_j^f) \quad . \qquad (20)$$

For Equation (20) we employed the fact that for the spike response of a presynaptic spike at time t_i^f, we can neglect the contribution of future postsynaptic spikes with $t_j^f > t_i^f$, and vice versa. Hence, similar to the Z_i and Z_j, the product Z_iZ_j can be written by means of the spike kernels ζ_{ij}:

$$Z_iZ_j = \sum_{t_i^f} Z_j(t_i^f)\zeta_{ij}(t - t_i^f) + \sum_{t_j^f} Z_i(t_j^f)\zeta_{ij}(t - t_j^f),$$

$$\zeta_{ij}(t) = e^{-\frac{t}{\tau_{z_{ij}}}} \Theta(t). \qquad (21)$$

In contrast to Z_i and Z_j, where all spikes have equal strength, for Z_iZ_j the spike response of each presynaptic spike t_i^f is scaled by the current value of the postsynaptic Z_j trace, respectively by $Z_i(t_j^f)$ for each postsynaptic spike t_j^f.

As the E_{ij} trace is just a low-pass filtered version of the product Z_iZ_j, we can analogously write the E_{ij} and P_{ij} traces as spike response models:

$$E_{ij}(t) = \sum_{t_i^f} Z_j(t_i^f)\alpha_{ij}(t - t_i^f) + \sum_{t_j^f} Z_i(t_j^f)\alpha_{ij}(t - t_j^f) \quad (22)$$

$$P_{ij}(t) = \sum_{t_i^f} Z_j(t_i^f)\pi_{ij}(t - t_i^f) + \sum_{t_j^f} Z_i(t_j^f)\pi_{ij}(t - t_j^f), \quad (23)$$

with spike response kernels:

$$\alpha_{ij}(t) = a_{ij}\left(e^{-\frac{t}{\tau_{z_{ij}}}} - e^{-\frac{t}{\tau_e}}\right)\Theta(t) \qquad (24)$$

$$\pi_{ij}(t) = a_{ij}\left[b_{ij}\left(e^{-\frac{t}{\tau_{z_{ij}}}} - e^{-\frac{t}{\tau_p^*}}\right) + c\left(e^{-\frac{t}{\tau_p^*}} - e^{-\frac{t}{\tau_e}}\right)\right]\Theta(t) (25)$$

We remark that Equations (20, 22, 23) are ambiguous for the limit case of simultaneous pre- and postsynaptic spikes ($t_i^f = t_j^f$), as it is unclear whether the sampled Z_i and Z_j correspond to the values directly before or after the spikes. This is resolved in Section 2.3.3.

2.3.3. BCPNN solution with reduced operations: analytical II

For the analytical update of the Z, E and P traces derived in Section 2.3.1, we observe that especially the update of P traces is expensive in terms of number of operations. In the presented BCPNN architecture, each MCU has approximately 10,000 inputs and correspondingly as many outputs. It would therefore be of great benefit to reduce the computational cost of the update of the synaptic traces. We achieve this by transforming the BCPNN variables to a new set of state variables that all decay exponentially over time and are only increased when a spike occurs. This is motivated by the spike response model formulation (Section 2.3.2), where the Z_i, E_i, P_i traces are superpositions of the spike response kernels ζ_i, α_i, and π_i, which in turn are linear combinations of the exponential functions $e^{-\frac{t}{\tau_{z_i}}}$, $e^{-\frac{t}{\tau_e}}$, and $e^{-\frac{t}{\tau_p^*}}$. Due to the linearity of the system we can choose these exponentials as new state variables to equally describe the BCPNN dynamics.

This second analytic solution of the BCPNN model is henceforth called *analytical II* in this paper.

2.3.3.1. Presynaptic traces.

For the presynaptic side, we introduce the new state variables Z_i^*, E_i^*, and P_i^*:

$$Z_i^*(t) = \sum_{t_i^f} \mathrm{e}^{-\frac{t-t_i^f}{\tau_{z_i}}} \Theta(t - t_i^f) \quad (26)$$

$$E_i^*(t) = \sum_{t_i^f} \mathrm{e}^{-\frac{t-t_i^f}{\tau_e}} \Theta(t - t_i^f) \quad (27)$$

$$P_i^*(t) = \sum_{t_i^f} \mathrm{e}^{-\frac{t-t_i^f}{\tau_p^*}} \Theta(t - t_i^f) \quad , \quad (28)$$

which can be used to express Z_i, E_i, and P_i:

$$Z_i(t) = Z_i^*(t) \quad (29)$$
$$E_i(t) = a_i \left(Z_i^*(t) - E_i^*(t)\right) \quad (30)$$
$$P_i(t) = a_i \left[b_i \left(Z_i^*(t) - P_i^*(t)\right) + c\left(P_i^*(t) - E_i^*(t)\right)\right] \quad . \quad (31)$$

The time course of the new state variables as a response to an example spike train is shown in **Figure 3C**. Note that we have introduced Z_i^* although it is identical to Z_i in order to emphasize the concept of the new representation with exponentially decaying state variables.

Instead of performing an event-based update of the original state variables Z_i, E_i, and P_i, we can update Z_i^*, E_i^*, and P_i^*: Given that there is no spike between t^{last} and t, the state evolves from t^{last} to t, with $\Delta t = t - t^{\text{last}}$, as:

$$Z_i^*(t) = Z_i^*(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_{z_i}}} + S_i(t) \quad (32)$$

$$E_i^*(t) = E_i^*(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_e}} + S_i(t) \quad (33)$$

$$P_i^*(t) = P_i^*(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}} + S_i(t) \quad (34)$$

Thus, between any two times we only have to calculate the exponential decay with τ_{z_i}, τ_e, and τ_p^*. At a new spike, we add 1 to all of the new state variables, compared to the classical lazy model, where only Z_i is increased (cf. **Figure 3**). Of course, equivalent new state variables and the same updating scheme can be used for the postsynapic side.

2.3.3.2. Synaptic traces. For updating the synaptic variables, an analogy can be made to the presynaptic traces. Again, we introduce new state variables E_{ij}^* and P_{ij}^*:

$$E_{ij}^*(t) = \sum_{t_i^f} Z_j(t_i^f) \mathrm{e}^{-\frac{t-t_i^f}{\tau_e}} \Theta(t - t_i^f) + \sum_{t_j^f} Z_i(t_j^f) \mathrm{e}^{-\frac{t-t_j^f}{\tau_e}} \Theta(t - t_j^f) \quad (35)$$

$$P_{ij}^*(t) = \sum_{t_i^f} Z_j(t_i^f) \mathrm{e}^{-\frac{t-t_i^f}{\tau_p^*}} \Theta(t - t_i^f) + \sum_{t_j^f} Z_i(t_j^f) \mathrm{e}^{-\frac{t-t_j^f}{\tau_p^*}} \Theta(t - t_j^f) \quad (36)$$

These, together with Z_i^* and Z_j^*, can be used to express E_{ij} and P_{ij}:

$$E_{ij}(t) = a_{ij} \left(Z_i^*(t) Z_j^*(t) - E_{ij}^*(t)\right) \quad (37)$$

$$P_{ij}(t) = a_{ij} \left[b_{ij} \left(Z_i^*(t) Z_j^*(t) - P_{ij}^*(t)\right) + c\left(P_{ij}^*(t) - E_{ij}^*(t)\right)\right] \quad (38)$$

We first consider the event-based update of the new synaptic state variables E_{ij}^* and P_{ij}^* for a presynaptic spike only (which is equivalent to a postsynaptic spike only). The case of simultaneous pre- and postsynaptic spikes is treated separately afterwards. In order to advance E_{ij}^* and P_{ij}^* from their last updated time t^{last} to t, with $\Delta t = t - t^{\text{last}}$ and no spike within this interval, the update goes as follow:

$$E_{ij}^*(t) = E_{ij}^*(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_e}} + S_i(t) \cdot Z_j(t) \quad (39)$$

$$P_{ij}^*(t) = P_{ij}^*(t^{\text{last}}) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau_p^*}} + S_i(t) \cdot Z_j(t), \quad (40)$$

i.e., E_{ij}^* and P_{ij}^* decay exponentially from their last states and, for the case of a presynaptic spike t_i^f at time t, increase by the sampled postsynaptic $Z_j(t)$ trace. Here lies the difference to the presynaptic update, where each spike has the same effect, whereas the synaptic E_{ij}^* and P_{ij}^* traces are increased depending on the current Z_j value of the postsynaptic side, as the synaptic traces keep track of the *overlap* of pre- and postsynaptic activity.

The case of concurrent pre- and postsynaptic spikes is not well defined in the formulas for E_{ij}^* and P_{ij}^* (Equations 35, 36) and in the spike response model formulation (Equations 22, 23). Therefore, we turn back to the product $Z_i Z_j$, which at simultaneous pre- and postsynaptic spikes is increased by

$$\Delta_{ij} = Z_i^+ Z_j^+ - Z_i^- Z_j^- \quad . \quad (41)$$

Here Z_i^- (Z_j^-) denotes the Z-trace before the evaluation of a presynaptic (postsynaptic) spike, and Z_i^+ (Z_j^+) after the evaluation:

$$Z_i^+ = Z_i^- + S_i \quad , \quad Z_j^+ = Z_j^- + S_j \quad , \quad (42)$$

where S_i (S_j) is only non-zero if there is a presynaptic (postsynaptic) spike at the current time. Inserting Equation (42) into Equation (41) yields

$$\Delta_{ij} = (Z_i^- + S_i)(Z_j^- + S_j) - Z_i^- Z_j^- \quad (43)$$
$$= S_i Z_j^- + S_j Z_i^- + S_i S_j \quad (44)$$
$$= S_i Z_j^- + S_j Z_i^+ \quad . \quad (45)$$

The increment Δ_{ij} not only describes the change of $Z_i Z_j$, but also applies to updates for the new synaptic traces E_{ij}^* and P_{ij}^*. Equation (44) can be used when both spikes are evaluated *synchronously*, Equation (45) when both spikes are evaluated *consecutively*, i.e., when first the presynaptic spike is processed (first summand), and afterwards the postsynaptic spike (second summand). For the event-based benchmark simulations (Sections 2.2.2 and 3.1.2), where all spikes are discretized to multiples of dt, the latter strategy for Δ_{ij} is used for the update in the synapse array: first all presynaptic spikes are evaluated, then all postsynaptic spikes.

2.3.3.3. Initialization of exponential state variables. This section explains how to set the initial values of the new state variables (Z^*, E^*, P^*) from a given set of Z, E, P traces. Therefore, we first shorten the transformation formula of P_i (Equation 31) with new coefficients as:

$$P_i = \lambda_{zi} Z_i^* + \lambda_{ei} E_i^* + \lambda_{pi} P_i^* \tag{46}$$

$$\lambda_{zi} = a_i b_i \quad , \quad \lambda_{ei} = -a_i c \quad , \quad \lambda_{pi} = a_i (c - b_i) \quad . \tag{47}$$

For brevity, we have left out the time dependence of the states. The equivalent simplification can be applied for the postsynaptic traces. Similarly, the synaptic traces (Equations 37, 38) can be written as

$$E_{ij} = a_{ij} \left(Z_i^* Z_j^* - E_{ij}^* \right) \tag{48}$$

$$P_{ij} = \lambda_{zij} Z_i^* Z_j^* + \lambda_{eij} E_{ij}^* + \lambda_{pij} P_{ij}^* \quad , \tag{49}$$

with coefficients

$$\lambda_{zij} = a_{ij} b_{ij} \quad , \quad \lambda_{eij} = -a_{ij} c \quad , \quad \lambda_{pij} = a_{ij} \left(c - b_{ij} \right) \quad . \tag{50}$$

To turn the set of Z, E, P variables into the new state variables (Z^*, E^*, P^*), the following reverse transformation holds:

$$Z_i^* = Z_i \tag{51}$$

$$E_i^* = -\frac{E_i}{a_i} + Z_i^* \tag{52}$$

$$P_i^* = \frac{1}{\lambda_{pi}} \left(P_i - \lambda_{zi} Z_i^* - \lambda_{ei} E_i^* \right) \tag{53}$$

Note that the transformation has to be performed in the above order. The synaptic values are set as follows:

$$E_{ij}^* = -\frac{E_{ij}}{a_{ij}} + Z_i^* Z_j^* \tag{54}$$

$$P_{ij}^* = \frac{1}{\lambda_{pij}} \left(P_{ij} - \lambda_{zij} Z_i^* Z_j^* - \lambda_{eij} E_{ij}^* \right) \tag{55}$$

2.4. BENCHMARKS

To validate our implementation of the BCPNN we used several benchmarks, targeting either simulation run time or accuracy. As infrastructure for the simulations we used a cluster with Intel®Xeon®CPU E5-2690 2.90 GHZ. All benchmarks were implemented in C++ and compiled with GCC 4.7.1. All simulations were single-threaded. The time constants and other BCPNN parameters used for the benchmarks are listed in **Table 1**.

2.4.1. Simulation run time

To compare the computational cost of the different update strategies, we simulated the synaptic dynamics of a full hypercolumn with 10,000 inputs and 100 MCUs, see **Figure 2**. For both pre- and postsynaptic units we use independent Poisson spike trains, which are pre-generated and then read from a file to the main program, so that equal spike trains are used for the different update strategies. The simulation runs for 10 s, the Poisson rate is swept over a range of 0.01–100 Hz. For each rate and update strategy we assess the execution time per simulated second as the average of 5 runs with different random seeds. Although independent Poisson spike trains for the pre- and postsynaptic units will not be the case in realistic BCPNN applications including learning and retrieval of patterns, they sufficiently model the probabilistic nature of the MCUs and are thus favorable compared to regular spike trains. In order to measure only the computational cost of the synaptic updates, the stochastic MCUs are not simulated in this benchmark. However, for a fair comparison of the update strategies, we calculate the support value s_j (Equation 3) for all postsynaptic units at each time step, so that all β_j are calculated at every time step, and the weights w_{ij} are computed whenever a spike of presynaptic unit i arrives.

Table 1 | Parameters used in the execution time and accuracy benchmarks.

Synapse model	
Parameters	$\tau_{z_i} = 10$ ms presynaptic Z trace time constant
	$\tau_{z_j} = 15$ ms postsynaptic Z trace time constant
	$\tau_e = 20$ ms E trace time constant
	$\tau_p = 1000$ ms P trace time constant
	$\kappa = 1$ learning rate
	$\epsilon = 0.001$ minimum activity

Note that the values represent only one possible parameter set. Plausible ranges for the time constants are given in the text (Section 2.1.1). The execution time is not affected by the choice of the parameters, but, of course, the accuracy results may change when using different parameters.

2.4.2. Accuracy comparison

As many published results are based on an explicit Euler method (see e.g., Johansson and Lansner, 2007; Berthet et al., 2012; Kaplan and Lansner, 2014), we compare this numerical method to an exact analytical one in Section 3.2.2. Furthermore, we investigate the influence of using fixed-point operands with different number of bits instead of floating point numbers with double precision. For this purpose, we implemented a single BCPNN synapse in C++ with templates allowing the comparison of different number formats, making use of an in-house developed fixed-point library.

As stimuli for the BCPNN synapse we generated pre- and postsynaptic spike trains according to a homogeneous Poisson process with rate r. For the accuracy benchmarks not only the update frequency is important but also that different dynamical ranges of the BCPNN variables can be triggered, which requires different levels of correlation. To achieve that, we follow Kuhn et al. (2003) and create pre- and postsynaptic Poisson spike trains that share a fraction of c correlated spike times. Therefore, we create one correlated Poisson spike train with rate $c \cdot r$, and two independent Poisson spike trains with rate $(1 - c) \cdot r$ for the pre- and postsynaptic side. The correlated spike times are then added to both independent spike trains. To avoid a systematic zero-lag between pre- and postsynaptic spike times, the correlated spike times of the postsynaptic side are jittered according to a Gaussian distribution with standard deviation $\sigma = 5$ ms.

We run multiple simulations to investigate the effects of the Euler method and fixed-point operands, respectively. For each accuracy setting, stimuli are generated using 11 correlation factors c ranging from 0 to 1 in intervals of 0.1. For each of the different correlation factors, 10 different seeds are used for the Poisson processes, resulting in 110 simulations per accuracy setting. The stimuli are generated with an average rate of 1 Hz and the duration of each simulation is 1000 s. For the Euler method, spike times are set to multiples of the time step to avoid time discretization errors (Henker et al., 2012). For fixed-point operands, spike times are generated with a resolution of 0.01 ms.

To assess the accuracy of the different implementations, we consider absolute errors e_{abs}:

$$e_{\text{abs}} = |x - \hat{x}| \quad , \tag{56}$$

where x denotes the exact value (analytical solution with floating point double precision) and $\hat{x}$ is the approximation (either the Euler solution or the analytical solution with fixed-point operands). By point wise comparing each of the state variables (e.g., $w_{ij}, P_{ij} \ldots$), the accuracy can be assessed. The mean absolute error $\overline{e_{\text{abs}}}$ is the average of the single absolute errors determined at each time of a pre- or postsynaptic spike over all simulation runs. The normalized mean absolute error (NMAE) is the mean absolute error divided by the range of observed values x:

$$\text{NMAE} = \frac{\overline{e_{\text{abs}}}}{x_{\max} - x_{\min}} \quad , \tag{57}$$

which allows to compare the accuracy of several variables with different scales.

3. RESULTS

The two analytic solutions for spike-based BCPNN derived in Section 2.3 allow an efficient event-driven simulation of BCPNNs. In Section 3.1 we investigate how this reduces the computational footprint of BCPNN learning both formally and empirically. Aiming also for a small memory footprint, we evaluate the use of fixed-point numbers for the storage of BCPNN state variables, and compare the introduced discretization errors with the errors caused by the fixed step size simulation with the Euler method (Section 3.2).

3.1. COMPARISON OF SIMULATION STRATEGIES

In this section we compare the computational efficiency of the two analytical solutions of the BCPNN equations against each other and to the commonly used fixed step size implementation with the Euler method. We also investigate the benefit of using look-up tables for exponential decays in the analytical II method.

3.1.1. Number of operations

We start with a formal comparison between the two analytical update solutions by counting the steps of calculation required for an event-based update in each representation. Therefore, we categorize the operation into three classes: ADD combines both additions and subtractions, MUL stands for multiplications and divisions, EXP for calculations of the exponential function, and LOG for the natural logarithm.

Table 2 lists the number of operations needed by the analytical I and analytical II methods for different tasks: For the update of the presynaptic state variables (Z_i, E_i, P_i resp. Z_i^*, E_i^*, P_i^*) at an incoming spike, most notably, the analytical II method requires 6 MUL and 3 ADD operations less than the analytical I method. Instead, when the P_i value is retrieved, e.g., to calculate the synaptic weight w_{ij}, the analytical I method requires zero operations, while the analytical II method requires 2 ADD and 3 MUL operations to calculate P_i from Z_i^*, E_i^*, and P_i^*. Here, the difference between the two strategies manifests: while the analytical II is more efficient when the states are updated, it requires additional operations to determine the original states. Nevertheless, when adding up the counts of both tasks (pre-update and retrieval of P_i), e.g., when β_j is updated after a postsynaptic spike, the analytical II is still much more efficient than the analytical I method.

Table 2 | Arithmetic operations per task for different analytical update methods.

Task	Operation	Analytical I	Analytical II
Pre-update		Equations (7–9)	Equations (32–34)
	ADD	7	3
	MUL	12	6
	EXP	3	3
Retrieve P_i			Equation (46)
	ADD	–	2
	MUL	–	3
Syn-update		Equations (12, 13)	Equations (39, 40)
	ADD	6	2
	MUL	13	5
	EXP	3	2
Retrieve P_{ij}			Equation (49)
	ADD	–	2
	MUL	–	4
Update of w_{ij} at pre-spike	ADD	22	14
	MUL	39	29
	EXP	9	8
	LOG	1	1
Update of β_j at post-spike	ADD	8	6
	MUL	12	9
	EXP	3	3
	LOG	1	1

ADD, additions and subtractions; MUL, multiplications and divisions; EXP, computations of exponential function; LOG, natural logarithm. The operation counts of the analytical I method correspond to optimized versions of the referenced formulas using pre-calculated coefficients and intermediate steps. The different tasks are further specified in the text.

Similar results are found for the update of the synaptic state variables (E_{ij}, P_{ij} resp. E^*_{ij}, P^*_{ij}), where the advantage of the analytical II over the analytical I is even larger, cf. **Table 2**. Again, the analytical II strategy needs additional steps of computation for the retrieval of P_{ij}. For the typical case of a presynaptic spike, where all traces and the weight are updated (task "update of w_{ij} after prespike") and which includes the retrieval of all P-traces, the analytical II requires considerably less operations than the analytical I method. Note that the speedup of the analytical II is even higher when processing a post-synaptic spike, as then the weight needs not be calculated and thus the P traces need not be retrieved.

If we consider an array of BCPNN synapses, as in a hypercolumn unit of the reduced cortex model (**Figure 2**), where the pre- and postsynaptic traces are handled at margins of the array, it is the update and retrieval of the *synaptic* BCPNN state variables that make up the majority of the calculations. Assuming equal mean firing rates for the pre- and postsynaptic units, the P_{ij} values need to be retrieved on average only at every second spike event. In that case, the analytical II method requires roughly half the number of basic arithmetic operations (ADD and MUL) of the analytical I method, but only slightly less calculations of the natural exponential function.

3.1.2. Simulation run time

As a complement to the formal comparison, we measured the simulation run time required to simulate the update of synapses of one HCU with 10,000 inputs and 100 outputs for the different update strategies. The results for a sweep over the Poisson firing rates of the inputs and outputs, which is described in detail in Section 2.4.1, are shown in **Figure 4A**. As expected, for the fixed step size simulation with explicit Euler method and $dt = 1$ ms the execution time depends only slightly on the spike frequency: It takes ≈ 2.4 s to simulate 1 s of the network for firing rates up to 10 Hz, only for higher rates the run time increases significantly, which can be attributed to the more frequent calculation of synaptic weights. In contrast, for the event-based methods the execution time strongly depends on the firing activity: For very low spike rates, there is a baseline computational cost that can be attributed to the calculation of all postsynaptic biases β_j and support values s_j (Equation 3) at every time step (cf. Section 2.4.1). For Poisson rates of 0.1 Hz and higher, the execution time scales linearly with the firing rate. The update strategy with reduced operations (analytical II, green curve) clearly outperforms the conventional analytical update (analytical I, blue curve). For a typical average firing rate of 1 Hz of MCUs in a HCU (cf. Lansner et al., 2014) the analytical II strategy is more than 3 times faster than the real-time dynamics of the model, while the analytical I update runs approximately at real time. We remark that we optimized the C++ code of the analytical II update as good as possible, while the analytical I code is not optimized to the end. Thus, the results of latter can not be taken as final and should rather be interpreted as an intermediary result.

We compare the run time of the event-based methods directly to the fixed step size simulation in **Figure 4B**. For low spiking activity, the event-based methods are up to 100 times faster than the fixed step size method. At 1 Hz the analytical II strategy (green curve) runs more than 8 times faster than the simulation with Euler. Only for firing rates higher than 20 Hz the fixed step size approach is competitive with, respectively faster than the analytical II method.

Additional results for a 0.1 ms time discretization are provided in Appendix A4, showing a much higher speedup of event-driven methods against the fixed step size method.

3.1.3. Look-up tables for exponential functions

In another simulation we investigated the benefit of using look-up tables (LUTs) for the exponential functions instead of computing the exponential at each event. This is motivated by the number of exponential decays calculated per update (cf. **Table 2**), as well as by a profiling of the implemented C++ program which shows that a huge amount of simulation time is spent in the computation of the exponential function. Look-up tables are especially

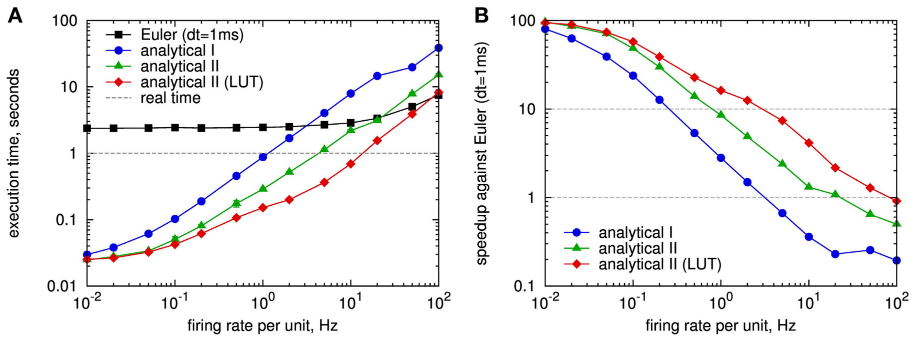


FIGURE 4 | Speed comparison of different simulation strategies for spike-based BCPNN: fixed step size simulation with explicit Euler method with 1 ms time step (Euler, black curve), event-driven simulation with analytical update (analytical I, cf. Section 2.3.1, blue) and analytical update with exponential state variables (analytical II, cf. Section 2.3.3), with and without using look-up tables (LUTs) for the exponential function (red, resp. green). **(A)** Execution time for simulating a full hypercolumn unit with 1 million BCPNN synapses for 1 s with different Poisson firing rates applied to both pre- and postsynaptic units of the HCU (cf. **Figure 2**). **(B)** Speedup of event-based simulation methods with respect to the fixed step size simulation with Euler method in **A**. Look-up tables were implemented for the exponential decay of type $\exp(-\frac{\Delta t}{\tau})$ for the time constants $\tau_{z_i}, \tau_{z_j}, \tau_e, \tau^*_p$. Each LUT had 3000 entries in steps of 1 ms.

beneficial in the used event-driven simulation (Section 2.2.2) where spike times are restricted to multiples of the time step dt. Calculations of the form

$$LUT(N, \tau, dt) = e^{-\frac{N \cdot dt}{\tau}} \tag{58}$$

are performed very often, where N is the number of time steps that have elapsed since the last update, and τ is one of the four involved time constants τ_{z_i}, τ_{z_j}, τ_e, τ_p^*. In a modified version of the analytical II implementation, we create look-up tables of Equation (58) for the four time constants, each with L entries for $N = 1 \ldots L$. Only if the number of elapsed time steps between two updates is larger than L, the exponential function Equation (58) is computed on demand.

The results for using look-up tables in the analytical II method are included in **Figure 4**: The implementation with look-up tables (red curve) speeds up the simulation for Poisson rates starting from 0.1 Hz, and is up to 3 times faster than the version without LUTs at 10 Hz spiking activity. Here, the size of the LUTs was chosen as $L = 3000$, covering update intervals up to 3 s, so that for a Poisson rate of 1 Hz on average 95 % of the inter spike intervals are handled by the look-up table. For the typical case of 1 Hz the LUT implementation is 1.9 times faster than the one without LUTs, 6.6 times faster than real time, and 16 times faster than the fixed step size simulation with explicit Euler method. For a wide spectrum of tested firing rates the analytical II solution with look-up tables is much more efficient than the fixed step size simulation with Euler, only for a firing rate of 100 Hz the latter performs slightly better (**Figure 4B**), so that in practical situations the fixed step size method becomes dispensable for the simulation of the abstract BCPNN cortex model.

3.2. FIXED-POINT NUMBERS FOR BCPNN TRACES AND THEIR ACCURACY

To store all traces of the 1 million BCPNN synapses of a full HCU, one requires more than 12 MB assuming a single precision floating point number occupying 4 byte for each state variable (Lansner et al., 2014). Targeting an implementation of the BCPNN model on neuromorphic hardware, the use of fixed-point numbers can reduce the number of computational and storage resources, possibly at the price of loosing precision. Therefore, we investigate the accuracy of using fixed-point operands to store the state variables in the event-based simulation with analytical II method, and compare it to the accuracy of the fixed step size simulation with the Euler method.

3.2.1. Value range estimation

For a fixed-point implementation of the BCPNN model, it is important to determine an upper bound of each state variable. This bound can be used to normalize the variables, so that an identical fixed-point representation can be used for all.

For a single exponential trace, be it the Z_i and Z_j traces in the standard analytical solution or the state variables of the analytical II solution, an upper bound can be calculated using a regular spike train with maximum rate $r_{\max}$. The value of this rate may be derived from the units' refractoriness period or as a multiple of the mean unit firing rate, accounting for short-term firing rate fluctuations. The upper bound can be calculated from the equilibrium state, where exponential decay and instantaneous increase at a spike equalize:

$$Z_i(t_n) \overset{!}{=} Z_i(t_{n+1}) = Z_i(t_n) \cdot e^{-1/(r_{\max} \cdot \tau_{z_i})} + S_i$$
$$\Rightarrow Z_{i,\max} = \frac{S_i}{1 - e^{-1/(r_{\max} \cdot \tau_{z_i})}} \tag{59}$$

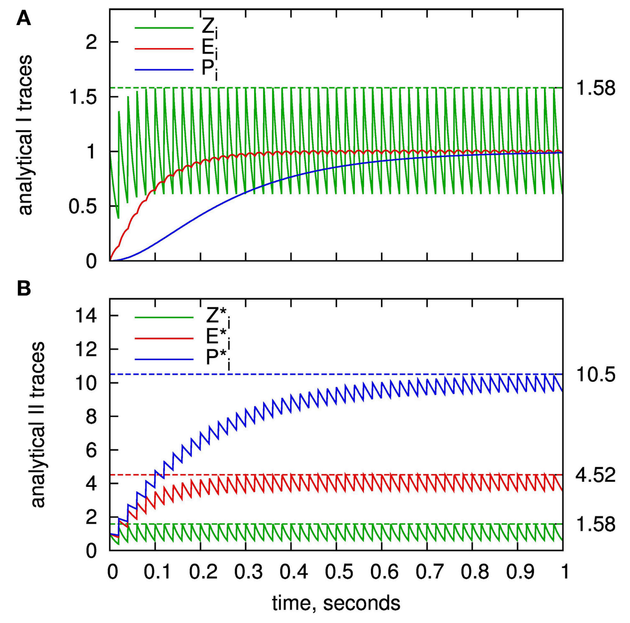


FIGURE 5 | Value range estimation of the BCPNN states variables: Traces of the *Z*, *E*, and *P* states for a regular spike train with frequency of 50 HZ. **(A)** Presynaptic traces according to the standard analytical solution (analytical I), as expressed in Equations (7–9). **(B)** Exponential traces of the improved analytical solution analytical II, Equations (32–34). Analytically calculated limits according to Equation (59) are plotted as dashed lines. Time constants used here: $\tau_{z_i} = 20\,\text{ms}, \tau_e = 80\,\text{ms}$ and $\tau_p^* = 200\,\text{ms}$.

The upper bounds of the presynaptic traces are illustrated in **Figure 5A**. They very closely match the actual maximums of the traces according to the employed regular spike train. For the traces of E_i and P_i, the same maximum as for Z_i can be used in good approximation. This is motivated from the differential equations of the model given in **Figure 1**: The worst-case assumption for Z_i from the maximum calculation would be a constant value of $Z_{i,\max}$. Given this input, the trace of E_i would approach the same value. The same argument in turn holds for P_i.

For the traces of the analytical II solution, Z_i^*, E_i^*, and P_i^*, Equation (59) can be used with according time constants. For $r_{\max} \cdot \tau \gg 1$, the maximum can be approximated as $r_{\max} \cdot \tau$ for an increment of $S_i = 1$. The highest absolute value is reached for the longest time constant, which is $\tau_p = 1000\,\text{ms}$ in our example parameter set. Assuming a refractoriness period of 1 ms, the worst-case upper bound would be $P_{i,\max}^* \approx 1000$. For a fixed-point implementation, a width of 10 integer bits would be sufficient to avoid any unwanted saturation or overflows. It can be expected that the actual maximum of P_i^* is significantly lower as it is extremely unlikely that a neuron (resp. a MCU) fires every 1 ms for a multitude of spikes. Thus, for a specific benchmark, a lower bound may be determined from simulation.

3.2.2. *Accuracy comparison*

We ran multiple simulations to investigate the effects of the Euler method and fixed-point operands, respectively. For each accuracy setting, a single BCPNN synapse was stimulated by pre- and postsynaptic Poisson spike trains of 1 Hz average rate. We applied different levels of correlation between pre- and postsynaptic spike trains in order to generate wide value ranges of the BCPNN variables, especially for w_{ij}. The simulation setup is described in detail in Section 2.4.2.

The accuracy results for both Euler method and fixed-point operands are shown in **Figure 6**. As accuracy measure, we assess the normalized mean absolute error (NMAE) as described in Section 2.4.2. To get an impression of the variable ranges in the simulations, we give their average, minimum and maximum in **Table 3**. Note that we only show the errors for w_{ij} and β_j (and not for the Z,E,P traces), as these are the only BCPNN variables that affect the activation of the postsynaptic units. As expected, the Euler method exhibits a linearly increasing accuracy with decreasing step size (**Figure 6A**). The accuracy is worse for the synaptic weight w_{ij} than for the bias β_j, as the w_{ij} error is affected by the errors of P_i,P_j, and P_{ij}, while β_j only depends on the accuracy of P_j. For 1 ms step size, which we used for the execution time benchmarks, the normalized mean absolute error of the synaptic weight lies far below 1 %. A reason for this relatively small error might be the exponentially decaying dynamics of the BCPNN variables, which keeps the accumulation of errors low.

For fixed-point operands, we used calculation with floating point precision, but quantized each intermediate result for a state variable to a fixed number of fractional bits. For the time constants and coefficients (Equations 47, 50) we used the highest available fixed-point precision (32 fractional bits) to minimize computational errors. This emulates the case that state variables are stored with limited precision to reduce storage space, but the arithmetic operations are designed such that they do not introduce additional numerical errors. Quantization errors can be modeled as a noise source with amplitude 2^{-b}, where b is the number of fractional bits. All errors scale according to this noise source (compare dashed line in **Figure 6B**). Again, the accuracy is higher for β_j than for w_{ij}, but now the ratio between w_{ij} and β_j errors is larger than in the Euler simulation.

Comparing these results answers the question what fixed-point operand resolution is required in our optimized analytical solution to achieve at least the same accuracy as state-of-the-art Euler methods. This can be derived from curves with equal mean absolute error, as shown in the lower diagram of **Figure 6C**. In terms of scaling, Euler method and fixed-point operands compare as

$$\overline{e_{\text{abs}}} = A_{\text{Euler}} \cdot dt = A_{\text{fixed}} \cdot 2^{-b} , \qquad (60)$$

where dt is the step size of the Euler method and $A_{\text{Euler}}, A_{\text{fixed}}$ are variable-specific constants. The corresponding line $dt = 2^{-b}$ is drawn as dashed line in the diagram. As expected from the previous results, the single errors follow this line, shifted by an offset. For a time step of $dt = 0.1$ ms 16 fractional bits or less are

Table 3 | Measured ranges of BCPNN state variables in accuracy simulations.

Variable	Mean	Min	Max
P_i	0.010	0.001	0.066
P_j	0.015	0.001	0.097
P_{ij}	0.0028	0.000	0.898
w_{ij}	1.57	−6.75	5.35
β_j	−4.38	−6.21	−2.32

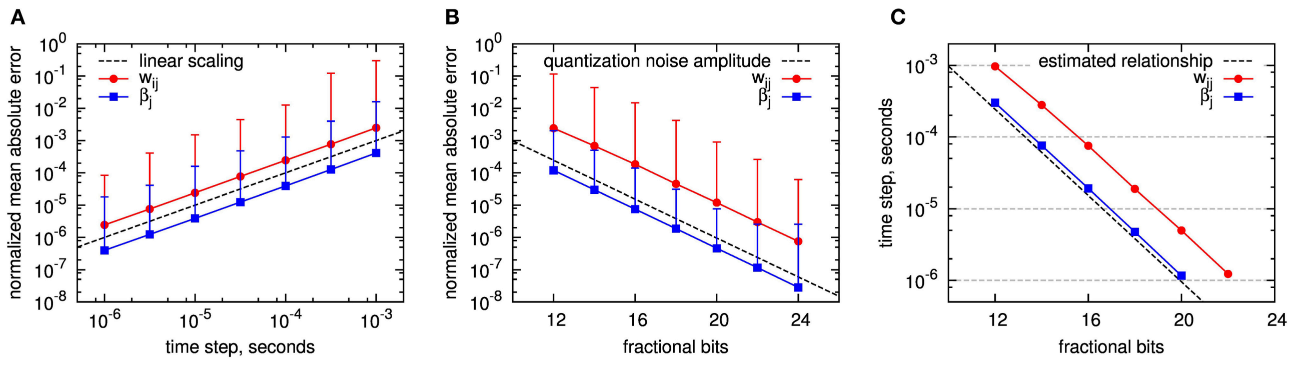


FIGURE 6 | Accuracy of fixed step size simulation with Euler method and event-driven analytic simulation using fixed point operands. The accuracy of w_{ij} and β_j is assessed by the normalized mean absolute error taken over a large set of experiments with the exact analytical solution as reference, see text for details. **(A)** Simulation with Euler, dependent on step size. The dashed line shows the linear scaling: $y(dt) = dt \cdot s^{-1}$. **(B)** Analytical solution with event-driven update (analytical II) using fixed-point representation with different counts of fractional bits. The dashed line shows the quantization noise amplitude: $y(b) = 2^{-b}$. The error bars in **A** and **B** denote the normalized maximum absolute error recorded within all simulations per setup. **(C)** Comparison between the errors introduced by the Euler method and the use of fixed-point numbers with limited number of fractional bits: For w_{ij} and β_j the location of *equal* mean absolute errors is plotted, depending on the step size for the Euler method, respectively the number of fractional bits of the fixed-point implementation. Dashed line: estimated relationship according to Equation (60).

required to achieve at least the same accuracy in all variables. A number of integer bits is required in addition to represent values greater than one. As shown in Section 3.2.1, a maximum of 10 integer bits is required in a worst-case scenario for the employed parameter set.

For the simulation of the reduced modular model of the cortex described in Section 2.1.2, for which a 1 ms time step seems to provide sufficient results (Lansner et al., 2014), only 12 fractional bits, and thus at maximum 22 bits in total, are needed to ensure equal or better accuracy compared to using the Euler method. Hereby, the required memory per state variable decreases by almost one third compared to using single precision floating point numbers.

Considering a 0.1 ms time step and a 64 bit floating point representation, which is commonly used in state-of-the art neural network simulations, fixed-point numbers with less than 32 bits yield competitive accuracy, so that the memory footprint reduces even by more than a half.

4. DISCUSSION

In this paper we derived two analytic solutions for the spike-based BCPNN learning rule. They enable an efficient event-driven simulation of spiking neural networks employing this learning rule, such as the reduced modular model of cortex (Lansner et al., 2014). The advantages of using an analytic over a fixed step size numeric solution are twofold: Firstly, it enables an event-driven update of the variables, and thereby significantly speeds up synaptic plasticity when interspike intervals are long compared to simulation time resolution. Secondly, it increases the precision of the calculations compared to fixed step size methods. Both aspects can be utilized for allocating resources in an existing hardware system efficiently or in conceiving a neuromorphic system based on the BCPNN computational paradigm.

4.1. CLASSIFICATION AND LIMITATIONS OF OPTIMIZATION

In our simulations including 1 million BCPNN synapses with pre- and postsynaptic activity at 1 Hz, we were able to reduce the execution time by a factor of 16 compared to the conventional fixed step size simulation with explicit Euler. One hypercolumn unit of the reduced cortex model was simulated more than 6 times faster than real time on a single CPU. Several factors are responsible for that speedup:

By employing the analytical I solution of the BCPNN model, the *event-driven simulation* becomes feasible and clearly defeats the time-driven simulation at the chosen working point of 1 Hz firing rate. In general, the event-driven approach is mostly advantageous over the time-driven approach when the firing rates are low and connectivity is sparse (Brette et al., 2007). Hence, as long as the inter-spike intervals are large compared to the simulation step size, the analytic event-driven simulation can effectively reduce the execution time of spiking neural networks, independent whether the BCPNN synapses connect single neurons or more abstract units like in the cortex model.

The *analytical II* solution requires on average only half of the basic arithmetic operations of the conventional analytical I solution for an event-based update, and slightly less calculations of the exponential function. Here, the computational cost is reduced by representing the same BCPNN dynamics with a set of exponentially decaying state variables, which is possible due to the linearity of the system. A similar approach has been taken by Brette (2006) for the exact simulation of leaky integrate-and-fire neurons with synaptic conductances, albeit with the restriction of equal excitatory and inhibitory synaptic time constants. Quite the opposite, the only limitation for the BCPNN synapse model is that the decay time constants of the three low pass filtering stages must differ. Nevertheless, when a specific network model requires equal time constants, one can still switch to the analytical I solution provided in Appendix A2, or try slightly different parameters.

The usage of *look-up tables* for the frequent calculation of exponential decays can further accelerate the simulation by a factor of 2 or 3. Precalculated look-up tables are a common tool in event-driven neural network simulations to reduce the cost for the calculation of complex functions (Brette, 2006; Ros et al., 2006). For BCPNN, LUTs for the exponential decay are beneficial as long as the time constants are homogeneous and do not vary from synapse to synapse. In our hybrid simulation of a hypercolumn unit, where spikes are discretized to multiples of the simulation step size, look-up tables not only accelerate the simulation, but also provide the same accuracy as the solution without LUTs. For simulations with arbitrary update intervals, linear interpolation can be used to achieve almost exact results (Brette, 2006). Alternatively, for the case of the exponential function, the computation can be split into two steps, e.g., by first retrieving the EXP separately for the integer and fractional bits of the exponent, and then multiplying the two obtained results. There remains the question for the optimal size and resolution of the look-up tables, which must be chosen depending on the used hardware platform (available memory, cache size) and the inter spike interval distributions of actual network models.

The optimizations presented in this paper focus on reducing the computational footprint for the spike-based BCPNN learning rule: In our benchmarks we have considered either a single synapse or an array of synapses, but not the dynamics of neurons or the MCUs. The efficient simulation of large recurrent networks with many HCUs entails many new issues, e.g., the distribution of hypercolumns across compute nodes and memory, the communication of spikes between HCU or the buffering of spikes, and gives rise to separate studies that are clearly out of scope of this paper.

4.2. ACCURACY

Fixed-point operands can reduce the memory footprint with the drawback of loosing precision compared to a floating point representation. To find the compromise between the two solutions, we assessed the accuracy of using fixed-point operands for the storage of the BCPNN state variables in an event-based simulation with the analytical II method (Section 3.2). The accuracy was compared to the errors introduced by the fixed step size simulation with explicit Euler method using 64 bit floating point numbers, which is commonly used in neural simulation. We found that fixed-point numbers with 22 bits assure equal or better accuracy for all BCPNN variables than the Euler method with 1 ms time step, resp. 26 bits for 0.1 ms time step.

The question remains about which accuracy is necessary in a practical situation. A previous study (Johansson and Lansner, 2004) on using fixed-point arithmetic for BCPNNs showed that an attractor network with 8 bit weights can offer the same storage capacity as an implementation with 32 bit floating point numbers. To achieve this, probabilistic fractional bits were used and the computation of the moving averages (low-pass filters) was performed in the logarithmic domain. Given these results, we speculate that also spike-based BCPNN can be implemented in fixed-point arithmetic with 16 or less bits without loosing computational capabilities, so that the required memory and memory bandwidth can be halved compared to 32 bit floating point numbers used in Lansner et al. (2014).

4.3. NEUROMORPHIC HARDWARE

Our optimizations can be directly incorporated for designing more efficient neuromorphic hardware systems. There are currently several diverse attempts for building large-scale hardware platforms, aiming for a more efficient simulation of large-scale neural models in terms of speed, power or scalability (Schemmel et al., 2012; Hasler and Marr, 2013; Benjamin et al., 2014; Furber et al., 2014; Merolla et al., 2014). As in our analysis, realizing synaptic plasticity is the most resource-demanding task, so that a focus of neuromorphic designs is in efficiently emulating plasticity mechanisms, most often implementing some variant of spike-timing dependent plasticity (STDP, Bi and Poo, 1998; Morrison et al., 2008) in analog or mixed-signal circuitry, see Azghadi et al. (2014) for a review.

An implementation of the BCPNN learning rule requires a stereotypical set of coupled low-pass filters, see **Figure 1**. Implementation of the rule in analog neuromorphic hardware is technically feasible, as there is large knowledge on building leaky integrators (Indiveri et al., 2011), and even the issue of long decay time constants in nanometer CMOS technologies can be resolved, e.g., with switched capacitor techniques (Noack et al., 2014). In this context, our optimized analytic solution offers an interesting alternative to the direct implementation of the original model equations: When using the analytical II solution, the stereotypical low-pass filters are only charged at incoming spikes, in contrast to the continuous coupling in a direct implementation. This alleviates the need for a continuous, variable-amplitude charging mechanism for the E and P traces. On the other hand, charging only at incoming spikes requires a more elaborate calculation of the output values, as present in the analytical II solution. However, this calculation needs to be performed only at spikes as well, allowing e.g., for an efficient implementation with switched-capacitor circuits.

The design of analog neuromorphic circuits is time-consuming and the circuits are affected by parameter variations due to device mismatch. Digital implementations are much less affected by these problems. They may be less energy and area efficient on the level of single elements and they do not allow for direct ion-channel-to-transistor analogies as employed in traditional neuromorphic designs (Hasler et al., 2007). However, they allow to fully utilize the energy efficiency and performance advantages of neural algorithms and modeling approaches, while offering better controllability and scalability.

Several purely digital neuromorphic systems support synaptic plasticity, implemented either on application-specific integrated circuits (Seo et al., 2011), on field-programmable gate arrays (FPGAs) (Cassidy et al., 2013) or a custom multiprocessor system using a larger number of general purpose ARM cores (SpiNNaker system, Furber et al., 2014). Recently Diehl and Cook (2014) showed how general STDP rules can be efficiently implemented on SpiNNaker, despite the system's restriction that synaptic weights can be modified only at the arrival of a presynaptic spike. By adopting their implementation of trace-based STDP, the event-driven spike-based BCPNN in variant analytical I or analytical II can be seamlessly integrated on the SpiNNaker hardware. As we do, Diehl and Cook (2014) use look-up tables for the exponential function; furthermore, SpiNNaker uses fixed-point arithmetic, so that our insights on the accuracy of fixed-point operands may find immediate application.

The event-driven approach is also amenable to state-of-the-art methods for reducing the energy of computation in digital systems. Recent multi-core hardware platforms support fine grained per-core power management, as for example demonstrated on the Tomahawk multiprocessor system-on-chip (MPSoC) architecture (Arnold et al., 2014; Noethen et al., 2014). By changing both the clock frequency and the core supply voltages of each processing element in a dynamic voltage and frequency scaling scheme (Höppner et al., 2012), the hardware performance can be adapted to the performance requirements to solve a particular part of the BCPNN in real time with reduced energy consumption, e.g., by regarding the number of incoming spikes per HCU per simulation step. In addition, within phases of low activity complete processing elements can be shut off to reduce leakage power consumption. Another candidate architecture for energy-efficient neural computation with BCPNNs is the multi-core Adapteva-Epiphany chip (Gwennup, 2011), which is optimized for power-efficient floating point calculations requiring only one fifth of the energy at equal flop rate as the state-of-the-art (but general-purpose) ARM's Cortex-A9 CPU.

Alternatively, spike-based BCPNN can be implemented on novel systems rather than on existing digital systems: For example, one may build dedicated digital hardware for the simulation of the BCPNN cortex model. Such a system containing compact supercomputer functionality can be prototyped in an FPGA with special units for the learning rule or the stochastic minicolumn units, and has therefore only low risk compared to mixed-signal implementations. Recently, Farahini et al. (2014) provided a concept for a scalable simulation machine of the abstract cortex-sized BCPNN model with an estimated power-dissipation of 6 kW in the technology of 2018, which is three orders of magnitudes smaller than for a full-cortex simulation on a supercomputer in comparable technology with 20 billion neurons and 10,000 times more synapses (see also Lansner et al., 2014). They assume the analytical I method for the event-driven updating of the BCPNN traces, and apply floating point units for arithmetic operations. Our work can further promote their performance: By using the analytical II method with look-up tables the computational cost can be further reduced; by moving to fixed-point arithmetics the required memory and memory bandwidth decreases, so that a low-power real-time simulation of the cortex becomes possible.

4.4. OUTLOOK

Of course, our optimizations can also be used to boost the simulation of spike-based BCPNN on conventional computing systems. For example, already the supercomputer simulations of the reduced cortex model by Benjaminsson and Lansner (2011) showed weak scaling and achieved the real-time operation when simulating one HCU per processor with the fixed step size Euler method ($dt = 1$ ms) and spike-based BCPNN synapses without E traces (the Z traces are directly passed to the P traces). Such large-scale BCPNN simulations are mostly bounded by computation rather than by inter-process communication (Johansson and Lansner, 2007; Lansner et al., 2014), as the firing activity is low and the connectivity is sparse and patchy. Hence, we conjecture that with our approach a speedup factor of 10 or more might be achieved. At the same time, our results can accelerate the simulations of small or medium-scale neural networks employing the spike-based BCPNN learning rule, with applications ranging from olfaction modeling (Kaplan and Lansner, 2014), reward learning (Berthet et al., 2012) to probabilistic inference (Tully et al., 2014). Regardless of whether the BCPNN is implemented in neuromorphic hardware, on a single PC or on supercomputers, the presented optimization through event-driven simulation with look-up tables can boost the success of the BCPNN paradigm as a generic plasticity algorithm in neural computation.

Furthermore, the BCPNN abstraction constitutes an alternative approach to tackle the energy efficiency wall for brain-sized simulations discussed in Hasler and Marr (2013): Instead of simulating every single neuron and synapse, one can choose a higher level of abstraction for the basic computational units in the brain (e.g., a minicolumn), use a powerful learning rule (e.g., spike-based BCPNN), and implement such networks in a lazy simulation scheme (e.g., on dedicated digital hardware), to finally achieve a very energy-efficient simulation of the brain.

AUTHOR CONTRIBUTIONS

Bernhard Vogginger, René Schüffny, Anders Lansner, Love Cederström, Johannes Partzsch, and Sebastian Höppner designed and conceived this work. René Schüffny and Bernhard Vogginger developed the analytical II solution. Bernhard Vogginger, Love Cederström, Johannes Partzsch, and Anders Lansner provided simulation or analysis code. Bernhard Vogginger, Love Cederström and Johannes Partzsch performed the experiments and analyzed the data. Bernhard Vogginger, René Schüffny, Anders Lansner, Love Cederström, Johannes Partzsch, and Sebastian Höppner wrote the paper and approved the final manuscript.

ACKNOWLEDGMENTS

We would like to thank Édi Kettemann for his work on the analytical I solution. This research was supported by the European Union Seventh Framework Programme (FP7) under Grant Agreement No. 604102 (Human Brain Project), and by the Swedish Science Council (VR-621-2012-3502) and SeRC (Swedish e-Science Research Centre).

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2015.00002/abstract

REFERENCES

Arnold, O., Matus, E., Noethen, B., Winter, M., Limberg, T., and Fettweis, G. (2014). Tomahawk: parallelism and heterogeneity in communications signal processing mpsocs. *ACM Trans. Embedded Comput. Syst.* 13, 107. doi: 10.1145/2517087

Azghadi, M., Iannella, N., Al-Sarawi, S., Indiveri, G., and Abbott, D. (2014). Spike-based synaptic plasticity in silicon: design, implementation, application, and challenges. *Proc. IEEE* 102, 717–737. doi: 10.1109/JPROC.2014.2314454

Bate, A., Lindquist, M., Edwards, I., Olsson, S., Orre, R., Lansner, A., et al. (1998). A Bayesian neural network method for adverse drug reaction signal generation. *Eur. J. Clin. Pharmacol.* 54, 315–321. doi: 10.1007/s002280050466

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J., et al. (2014). Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 699–716. doi: 10.1109/JPROC.2014.2313565

Benjaminsson, S., and Lansner, A. (2011). "Extreme scaling of brain simulations on JUGENE," in *Jülich Blue Gene/P Extreme Scaling Workshop* number FZJ-JSC-IB-2011-02, eds B. Mohr and W. Frings (Jülich: Forschungszentrum Jülich, Jülich Supercomputing Centre).

Berthet, P., Hellgren-Kotaleski, J., and Lansner, A. (2012). Action selection performance of a reconfigurable basal ganglia inspired model with Hebbian–Bayesian Go-NoGo connectivity. *Front. Behav. Neurosci.* 6:65. doi: 10.3389/fnbeh.2012.00065

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

Binzegger, T., Douglas, R. J., and Martin, K. A. (2007). Stereotypical bouton clustering of individual neurons in cat primary visual cortex. *J. Neurosci.* 27, 12242–12254. doi: 10.1523/JNEUROSCI.3753-07.2007

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., et al. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *J. Comput. Neurosci.* 23, 349–398. doi: 10.1007/s10827-007-0038-6

Brette, R. (2006). Exact simulation of integrate-and-fire models with synaptic conductances. *Neural Comput.* 18, 2004–2027. doi: 10.1162/neco.2006.18.8.2004

Cassidy, A. S., Georgiou, J., and Andreou, A. G. (2013). Design of silicon brains in the nano-cmos era: spiking neurons, learning synapses and neural architecture optimization. *Neural Netw.* 45, 4–26. doi: 10.1016/j.neunet.2013.05.011

Diehl, P. U., and Cook, M. (2014). "Efficient implementation of STDP rules on SpiNNaker neuromorphic hardware," in *Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN)* (Beijing).

Farahini, N., Hemani, A., Lansner, A., Clermidy, F., and Svensson, C. (2014). "A scalable custom simulation machine for the Bayesian confidence propagation neural network model of the brain," in *ASP-DAC* (Suntec City), 578–585.

Fiebig, F., and Lansner, A. (2014). Memory consolidation from seconds to weeks: a three-stage neural network model with autonomous reinstatement dynamics. *Front. Comput. Neurosci.* 8:64. doi: 10.3389/fncom.2014.00064

Furber, S., Galluppi, F., Temple, S., and Plana, L. (2014). The SpiNNaker project. *Proc. IEEE* 102, 652–665. doi: 10.1109/JPROC.2014.2304638

Gerstner, W., and Kistler, W. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, UK: Cambridge University Press.

Gwennup, L. (2011). Adapteva: more flops, less watts: epiphany offers floating-point accelerator for mobile processors. *Microprocess. Rep.* 2, 1–5. Available online at: http://www.adapteva.com/wp-content/uploads/2012/08/adapteva_mpr.pdf

Höppner, S., Shao, C., Eisenreich, H., Ellguth, G., Ander, M., and Schüffny, R. (2012). "A power management architecture for fast per-core DVFS in heterogeneous MPSoCs," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on* (Seoul), 261–264.

Hasler, J., and Marr, B. (2013). Finding a roadmap to achieve large neuromorphic hardware systems. *Front. Neurosci.* 7:118. doi: 10.3389/fnins.2013.00118

Hasler, P., Kozoil, S., Farquhar, E., and Basu, A. (2007). "Transistor channel dendrites implementing hmm classifiers," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on* (New Orleans, LA: IEEE), 3359–3362.

Henker, S., Partzsch, J., and Schüffny, R. (2012). Accuracy evaluation of numerical methods used in state-of-the-art simulators for spiking neural networks. *J. Comput. Neurosci.* 32, 309–326. doi: 10.1007/s10827-011-0353-9

Houzel, J.-C., Milleret, C., and Innocenti, G. (1994). Morphology of callosal axons interconnecting areas 17 and 18 of the cat. *Eur. J. Neurosci.* 6, 898–917. doi: 10.1111/j.1460-9568.1994.tb00585.x

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Johansson, C., and Lansner, A. (2004). *BCPNN Implemented with Fixed-Point Arithmetic.* Technical Report TRITA-NA-P0403, Royal Institute of Technology, Department of Numerical Analysis and Computer Science, Stockholm.

Johansson, C., and Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Netw.* 20, 48–61. doi: 10.1016/j.neunet.2006.05.029

Johansson, C., Sandberg, A., and Lansner, A. (2001). *A Capacity Study of a Bayesian Neural Network with Hypercolumns.* Technical Report TRITA-NA-P0120, Royal Institute of Technology, Department of Numerical Analysis and Computer Science. Stockholm.

Johansson, C., Raicevic, P., and Lansner, A. (2003). "Reinforcement learning based on a bayesian confidence propagating neural network," in *SAIS-SSLS Joint Workshop* (Örebro).

Kaplan, B. A., and Lansner, A. (2014). A spiking neural network model of self-organized pattern recognition in the early mammalian olfactory system. *Front. Neural Circuits* 8:5. doi: 10.3389/fncir.2014.00005

Kuhn, A., Aertsen, A., and Rotter, S. (2003). Higher-order statistics of input ensembles and the response of simple model neurons. *Neural Comput.* 15, 67–101. doi: 10.1162/089976603321043702

Lansner, A., and Ekeberg, Ö. (1989). A one-layer feedback artificial neural network with a bayesian learning rule. *Int. J. Neural Syst.* 1, 77–87. doi: 10.1142/S0129065789000499

Lansner, A., and Holst, A. (1996). A higher order Bayesian neural network with spiking units. *Int. J. Neural Syst.* 7, 115–128. doi: 10.1142/S0129065796000816

Lansner, A., Marklund, P., Sikström, S., and Nilsson, L.-G. (2013). Reactivation in working memory: an attractor network model of free recall. *PloS ONE* 8:e73776. doi: 10.1371/journal.pone.0073776

Lansner, A., Hemani, A., and Farahini, N. (2014). "Spiking brain models: computation, memory and communication constraints for custom hardware implementation," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)* (Suntec City: IEEE), 556–562.

Lansner, A. (2009). Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations. *Trends Neurosci.* 32, 178–186. doi: 10.1016/j.tins.2008.12.002

Lennie, P. (2003). The cost of cortical computation. *Curr. Biol.* 13, 493–497. doi: 10.1016/S0960-9822(03)00135-0

Lindquist, M., Stahl, M., Bate, A., Edwards, I., and Meyboom, R. (2000). A retrospective evaluation of a data mining approach to aid finding new adverse drug reaction signals in the who international database. *Drug Saf.* 23, 533–542. doi: 10.2165/00002018-200023060-00004

Lundqvist, M., Compte, A., and Lansner, A. (2010). Bistable, irregular firing and population oscillations in a modular attractor memory network. *PLoS Comput. Biol.* 6:e1000803. doi: 10.1371/journal.pcbi.1000803

Lundqvist, M., Herman, P., and Lansner, A. (2011). Theta and gamma power increases and alpha/beta power decreases with memory load in an attractor network model. *J. Cogn. Neurosci.* 23, 3008–3020. doi: 10.1162/jocn/a/00029

Lundqvist, M., Herman, P., and Lansner, A. (2013). Effect of prestimulus alpha power, phase, and synchronization on stimulus detection rates in a biophysical attractor network model. *J. Neurosci.* 33, 11817–11824. doi: 10.1523/JNEUROSCI.5155-12.2013

Meli, C., and Lansner, A. (2013). A modular attractor associative memory with patchy connectivity and weight pruning. *Network* 24, 129–150. doi: 10.3109/0954898X.2013.859323

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642

Morrison, A., Diesmann, M., and Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biol. Cybern.* 98, 459–478. doi: 10.1007/s00422-008-0233-1

Noack, M., Krause, M., Mayr, C., Partzsch, J., and Schüffny, R. (2014). "VLSI implementation of a conductance-based multi-synapse using switched-capacitor circuits," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on* (Beijing: IEEE), 850–853.

Noethen, B., Arnold, O., Perez Adeva, E., Seifert, T., Fischer, E., Kunze, S., et al. (2014). "A 105GOPS 36mm2 heterogeneous SDR MPSoC with energy-aware dynamic scheduling and iterative detection-decoding for 4G in 65nm CMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International* (San Francisco, CA), 188–189.

Ros, E., Carrillo, R., Ortigosa, E. M., Barbour, B., and Agís, R. (2006). Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics. *Neural Comput.* 18, 2959–2993. doi: 10.1162/neco.2006.18.12.2959

Sandberg, A., Lansner, A., Petersson, K., and Ekeberg, O. (2000). A palimpsest memory based on an incremental Bayesian learning rule. *Neurocomputing* 32-33, 987–994. doi: 10.1016/S0925-2312(00)00270-8

Sandberg, A., Tegnér, J., and Lansner, A. (2003). A working memory model based on fast hebbian learning. *Network* 14, 789–802. doi: 10.1088/0954-898X/14/4/309

Schemmel, J., Grübl, A., Hartmann, S., Kononov, A., Mayr, C., Meier, K., et al. (2012). "Live demonstration: a scaled-down version of the BrainScaleS wafer-scale neuromorphic system," in *Proceedings of the 2012 IEEE International Symposium on Circuits and Systems (ISCAS)* (Seoul), 702.

Seo, J.-S., Brezzo, B., Liu, Y., Parker, B. D., Esser, S. K., Montoye, R. K., et al. (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* (San Jose, CA: IEEE), 1–4.

Silverstein, D. N., and Lansner, A. (2011). Is attentional blink a byproduct of neocortical attractors? *Front. Comput. Neurosci.* 5:13. doi: 10.3389/fncom.2011.00013

Tully, P. J., Hennig, M. H., and Lansner, A. (2014). Synaptic and nonsynaptic plasticity approximating probabilistic inference. *Front. Synaptic Neurosci.* 6:8. doi: 10.3389/fnsyn.2014.00008

Wahlgren, N., and Lansner, A. (2001). Biological evaluation of a hebbian–bayesian learning rule. *Neurocomputing* 38, 433–438. doi: 10.1016/S0925-2312(01)00370-8

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 22 September 2014; accepted: 03 January 2015; published online: 22 January 2015.
*Citation: Vogginger B, Schüffny R, Lansner A, Cederström L, Partzsch J and Höppner S (2015) Reducing the computational footprint for real-time BCPNN learning. Front. Neurosci. **9**:2. doi: 10.3389/fnins.2015.00002*
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.

REVIEW ARTICLE
published: 02 March 2015
doi: 10.3389/fnins.2015.00051

Plasticity in memristive devices for spiking neural networks

Sylvain Saïghi[1]*, Christian G. Mayr[2], Teresa Serrano-Gotarredona[3], Heidemarie Schmidt[4], Gwendal Lecerf[1], Jean Tomas[1], Julie Grollier[5], Sören Boyn[5], Adrien F. Vincent[6], Damien Querlioz[6], Selina La Barbera[7], Fabien Alibart[7], Dominique Vuillaume[7], Olivier Bichler[8], Christian Gamrat[8] and Bernabé Linares-Barranco[3]

[1] *Laboratoire d'Intégration du Matériau au Système, UMR CNRS 5218, Université de Bordeaux, Talence, France*
[2] *Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland*
[3] *Instituto de Microelectrónica de Sevilla, IMSE-CNM, Universidad de Sevilla and CSIC, Sevilla, Spain*
[4] *Faculty of Electrical Engineering and Information Technology, Technische Universität Chemnitz, Chemnitz, Germany*
[5] *Unité Mixte de Physique CNRS/Thales, Palaiseau, France Associated to University Paris-Sud, Orsay, France*
[6] *Institut d'Electronique Fondamentale, Université Paris-Sud, CNRS, Orsay, France*
[7] *Institut d'Electronique, Microelectronique et Nanotechnologies, UMR CNRS 8520, Villeneuve d'Ascq, France*
[8] *CEA, LIST, Saclay Nano-INNOV PC 172, Gif sur Yvette, France*

Edited by:
Themis Prodromakis, University of Southampton, UK

Reviewed by:
Shimeng Yu, Arizona State University, USA
Doo Seok Jeong, Korea Institute of Science and Technology, South Korea
Ronald Tetzlaff, TU Dresden, Germany

****Correspondence:***
Sylvain Saïghi, Laboratoire d'Intégration du Matériau au Système, UMR CNRS 5218, Université de Bordeaux, 351 Cours de la Libération, 33405 Talence, France
e-mail: sylvain.saighi@ims-bordeaux.fr

Memristive devices present a new device technology allowing for the realization of compact non-volatile memories. Some of them are already in the process of industrialization. Additionally, they exhibit complex multilevel and plastic behaviors, which make them good candidates for the implementation of artificial synapses in neuromorphic engineering. However, memristive effects rely on diverse physical mechanisms, and their plastic behaviors differ strongly from one technology to another. Here, we present measurements performed on different memristive devices and the opportunities that they provide. We show that they can be used to implement different learning rules whose properties emerge directly from device physics: real time or accelerated operation, deterministic or stochastic behavior, long term or short term plasticity. We then discuss how such devices might be integrated into a complete architecture. These results highlight that there is no unique way to exploit memristive devices in neuromorphic systems. Understanding and embracing device physics is the key for their optimal use.

Keywords: memristive device, memristor, neuromorphic engineering, plasticity, hardware neural network

INTRODUCTION

In 1971, Leon Chua indicated the possible existence of a fourth basic electrical component (Chua, 1971). This component, the memristor, would complement those already known namely resistance, capacitor, and inductor, and offer new opportunities for system design (Chua and Kang, 1976). In particular, Chua proposed to use memristors or similar memristive devices to fabricate synapses and neurons following the Hodgkin–Huxley formalism. From this theoretical work, several publications have cited the memristive phenomenon without naming it as such and without linking it to Chua's theory (Upadhyaya and Chandra, 1995; Lau et al., 2004; Waser and Aono, 2007; Wu et al., 2007; Pershin and Di Ventra, 2008). HP labs were the first to recognize a device as a memristor in 2008 (Strukov et al., 2008), and they highlighted both the technology and its possible applications.

In parallel, the designers of the neuromorphic community worked hard on achieving CMOS neurons to reach electrical energy consumption of the order of picojoule per spike (Wijekoon and Dudek, 2008; Livi and Indiveri, 2009; Rangan et al., 2010; Merolla et al., 2011; Joubert et al., 2012). However, if the neuron implementation still have to face important challenges to match the neurons density and functionality required for neuromorphic circuits, the most abundant element in a neural network is the synapse. Consequently, most of the efforts have been concentrated on achieving high density memories with embedded synaptic functionalities (i.e., synaptic plasticity) in a single component. To become functional, the realization of a plastic synapse requires three parts: (i) synaptic weight storage, (ii) circuit for updating this weight depending on the network activity, and (iii) circuit for information transmission between two neurons. The neuromorphic community has developed a strong interest in memristive devices because these nanodevices and the associated integration strategies offer potential solutions to realize these three functions.

Resistive Random Access Memory (ReRAM) technologies in its broad sense have been developed for pure memory applications but can fall into the memristive system classification (Baek et al., 2004; Lee et al., 2008; Wong et al., 2012). These different technologies are mostly used in binary mode and are at the stage of industrialization and commercialization (e.g., ReRAM from Panasonic and Samsung) with high endurance, low energy, and high integration capability performances (Kawahara et al., 2012; Liu et al., 2013). Such performances can be an interesting platform for the implementation of synaptic weight storage (even in binary mode) if integration strategies and specific architectures are developed in order to offer a suitable solution to the large access required between neuron (i.e., computing node) and synapses (memory) inherent to parallel computing in neuromorphic circuits (and unsolved by Von Neumann architectures and associated bottleneck). In addition, their use in analog mode (or multilevel), is the subject of great attention and could be an effective solution for the implementation of synaptic functions.

Defining a memristor itself (see **Figure 1A**) can be debatable. Leon Chua now defines a memristor as any element that has an I(V) curve pinched at 0 V (**Figure 1B**) (Chua, 2014). This definition is widely used in the literature for characterizing devices, and in this paper we synonymously use the historic word memristor or the more generic "memristive device." A general feature of memristive devices is to offer a non-volatile modification of its resistance (or conductance) as a function of the current (charge) or voltage (flux) driving the device. In particular, neuromorphic circuit designers prefer to think of memristors as resistive components that have the following properties: (i) the greater the electrical charge that has passed through the component, the more the resistance value decreases, (ii) the resistance value is stored in the element even after it is turned off. Moreover, this modification appears if the charge through the memristor goes over a "threshold" (**Figure 2D**).

Memristors can be realized using several technologies and we can categorize these technologies in four large families. The first includes anionic and cationic Red-Ox devices operating on Oxidation–Reduction principles. The second is phase-change memories (PCM), where resistive switching is connected with a physical phase change. Organic elements represent the third family. The fourth family finally comprises elements using purely electronic effects such as ferroelectric tunnel and spintronic memristors. These technologies possess different behaviors and therefore different fields of application. As part of this paper about synaptic plasticity, we also point out that these technologies will lead to different plastic behaviors and learning rules. These differences enrich the palette of possibilities for neuromorphic design. As Jeong et al. (2013), the purpose of this paper is not to present an exhaustive list of memristive technology and of their associated behavior, but rather to present the different forms of learning that have been observed. In our paper, all data about memristive devices have been measured by at least one of the co-authors.

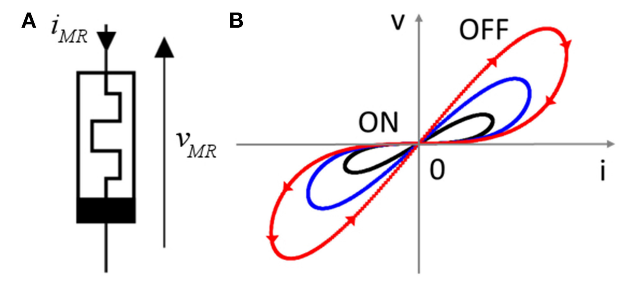


FIGURE 1 | (A) Symbol of memristor; **(B)** characteristic transport features of memristors: pinched iv loops for different values of the maximum injected current.

If computing and memorization principles in neural networks are not completely understood, it is now widely recognized that learning in such systems is associated to synaptic weight modification that tends to reinforce or depress the strength of the connection between two neurons and grouped into the wide class of synaptic plasticity. The most popular description of learning was proposed by Hebb with the postulate "who fire together, wire together" (Hebb, 1949). In other words, two neurons presenting a correlated activity will tend to reinforce their synaptic connection. A first requirement is to define what we call neuron activity: two different approaches are commonly used, (i) rate coding strategies correspond to the definition of neuron activity as the mean firing rate estimated on a chosen time window while (ii) temporal coding corresponds to the assignment of neuron activity to a single spike event with a given time stamp with respect to the other spiking neurons considered in the network. Based on this different coding strategies, variations of Hebbian learning have been proposed such has Spike Rate Dependent Plasticity (SRDP) or the very popular Spike Timing Dependent Plasticity (STDP). In particular, STDP has attracted a large interest in the memristive device community because of its practical implementation based on overlapping pulses coming from the pre and post neurons. We present in Section STDP Learning Thanks to Overlapping Events theoretical elements that allow the understanding of the application of this basic learning algorithm. Starting from this ideal case, we present practical implementations of STDP in solid state devices and show how material constraint (i.e., switching mechanism, operating conditions, ...) can be used to realize various form of STDP. Then we present two cases of "ferroelectric" memristors based on thin film semiconductor-metal-metaloxide compounds. These compounds were some of the first materials to be used as memristive synapses (see Kuzum et al., 2013 for a review). The first of our ferroelectric memristors is based on several 100 nm thick $BiFeO_3$ films experiencing resistive switching in the Schottky barrier formed with one of the contacts. Specifically, the memristive effect in these devices is effected by a change of the depletion layer of the Schottky diode due to a non-volatile charge transfer similar to the "moving barrier" of TiO_2. The second consists of ferroelectric tunnel junctions of very thin (~1 nm) $BiFeO_3$ films in which tunneling resistance is linked to the polarization of the barrier. They differ radically by the time scales on which they operate and thus by the contexts in which they could be used. A third case based on spin-transfer torque magnetic tunnel junction is also presented in Section Spin-Transfer Torque Magnetic Tunnel Junction as a Stochastic Synapse. It presents a stochastic behavior in learning which is in some ways reminiscent of biological neural networks. In Section SRDP with Memristive Devices, we present different form of SRDP observed in biological synapses and of interest for spike rate coding strategies. We first show how Short Term Plasticity, corresponding to a temporary modification of the weight that tends to relax toward a resting state, can be used to implement rate dependent modification of the weight. A second example describes how Short Term/Long Term plasticity transitions can be reproduced by taking advantage of device stability characteristics. Before the conclusion, Section Toward Memristor-CMOS Architectures and Circuits opens the discussion on the characteristics of circuit

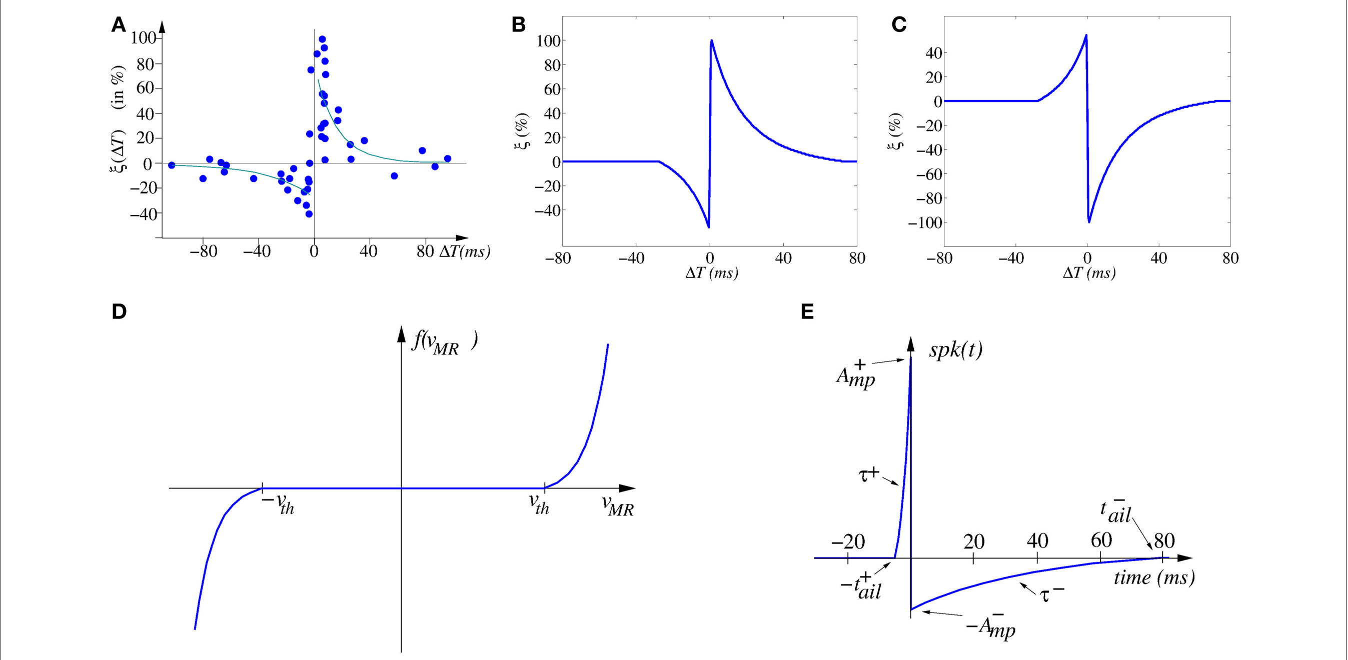


FIGURE 2 | (A) Experimentally measured STDP function $\xi_i(\Delta T)$ on biological synapses (data from Bi and Poo, 1998, 2001). **(B)** Ideal STDP update function used in computational models of STDP synaptic learning. **(C)** Anti-STDP learning function for inhibitory STDP synapses. **(D)** Shape of memristor weight update function $f(v_{MR})$. **(E)** Spike shape waveform.

architectures that will drive memristors following their electrical behavior.

STDP LEARNING THANKS TO OVERLAPPING EVENTS

THEORETICAL PRINCIPLES

STDP is the ability of natural or artificial synapses to change their strength according to the precise timing of individual pre- and/or post-synaptic spikes (Gerstner et al., 1993, 1996; Markram et al., 1997; Bi and Poo, 1998, 2001; Zhang et al., 1998; Feldman, 2000; Mu and Poo, 2006; Cassenaer and Laurent, 2007; Jacob et al., 2007; Young, 2007; Finelli et al., 2008; Masquelier et al., 2008, 2009). A comprehensive overview of STDP and of its history can be found elsewhere (Sjöström and Gerstner, 2010). STDP learning in biology is inherently asynchronous and on-line, meaning that synaptic incremental update occurs while neurons and synapses transmit spikes and perform computations in parallel. Early proposals of this used artificial time-multiplexing to alternate continuously and synchronously between “performing” and “weight update” phases (Snider, 2008), thus requiring global system-wide synchronization. This can become a severe handicap when scaling up systems. Another option is a fully asynchronous implementation for memristor-based STDP where “performing” and “weight update” phases happen simultaneously in a natural manner, as in biology (Linares-Barranco and Serrano-Gotarredona, 2009a,b; Zamarreño-Ramos et al., 2011; Bichler et al., 2012b; Kuzum et al., 2012), and where there is no need for any global synchronization.

Figure 2A shows the change of synaptic strength (in percent) measured experimentally from biological synapses as function of relative timing $\Delta T = t_{\text{pos}} - t_{\text{pre}}$ between the arrival time t_{pre} of a pre-synaptic spike and the time t_{pos} of the generation of a post-synaptic spike. Although the data shows stochasticity, we can infer an underlying interpolated function $\xi(\Delta T)$ as shown in **Figure 2B**.

$$\xi(\Delta T) = \begin{cases} a^{+}e^{-\frac{\Delta T}{\tau^{+}}} & if\ \Delta T > 0 \\ -a^{-}e^{-\frac{\Delta T}{\tau^{-}}} & if\ \Delta T < 0 \end{cases} \tag{1}$$

For a causal pre- to post-spike timing relation ($\Delta T > 0$) the strength of the synapse is increased, while for an anti-causal relation ($\Delta T < 0$) it is decreased. In the case of synapses with negative synaptic strength (as in some artificial realizations), the reversed version shown in **Figure 2C** can be used. Microchip CMOS circuit implementations of STDP rules that follow the description of Equation (1) have been reported (Indiveri et al., 2006), which result in about 30 transistors per plastic synapse, and thus may lead to high costs for their hardware realization. There is, overall, general thinking that STDP is very expensive to implement in conventional CMOS microchips (Fieres et al., 2008; Khan et al., 2008). However, it can be implemented with just one memristor per synapse if appropriate peripheral signal conditioning neurons are used in hybrid CMOS/memristor realizations.

For our purpose, we will consider a particular type of memristors, named voltage/flux driven memristor, which can be mathematically defined by.

$$\begin{aligned} i_{MR} &= G(w, v_{MR})\, v_{MR} \\ \dot{w} &= f(v_{MR}) \end{aligned} \tag{2}$$

Memristor current and voltage are in general related through a non-linear conductance G (in the i_{MR} vs. v_{MR} plane), whose shape is tuned by parameter w. Most of the times, however, we may approximate the conductance as being totally linear $i_{MR} = G(w)v_{MR}$, where the value of w is dependent on the history of v_{MR}. Parameter w represents some structural property of the memristor. This parameter changes non-linearly as a function $f(\ )$ of the evolution of the memristor voltage v_{MR}, so that the derivative of w is governed by the second equation in (Equation 2). A typical shape of this function is shown in **Figure 2D**, where a "dead zone" between two threshold voltages is present. While the memristor voltage is kept within this dead zone, parameter w will remain constant, and G will not change. But if the memristor voltage goes out of the dead zone, the (linear or non-linear conductance G) will change.

The STDP learning rule (as modeled by Equation 1) can, in theory, be implemented by (i) using a particular type of voltage/flux driven memristor (Jo et al., 2010), while (ii) providing appropriately shaped pre- and post-synaptic spikes available at both synapse (memristor) electrodes (Zamarreño-Ramos et al., 2011). For example, we can consider a pair of identical pre- and post-synaptic spikes with a shape resembling that of biological spikes (see **Figure 2E**), with an on-set duration $|t^+_{ail}|$ and a tail of duration $|t^-_{ail}|$,

$$spk(t) = \begin{cases} A^+_{mp} \frac{e^{\frac{t}{\tau^+}} - e^{-\frac{t^+_{ail}}{\tau^+}}}{1 - e^{-\frac{t^+_{ail}}{\tau^+}}} & if\ -t^+_{ail} < t < 0 \\ -A^-_{mp} \frac{e^{-\frac{t}{\tau^-}} - e^{-\frac{t^-_{ail}}{\tau^-}}}{1 - e^{-\frac{t^-_{ail}}{\tau^-}}} & if\ 0 < t < t^-_{ail} \\ 0 & if\ otherwise \end{cases} \tag{3}$$

Under these circumstances, memristor voltage is $v_{MR}(t, \Delta t) = \alpha_{pos}\ spk(t) - \alpha_{pre}\ spk(t + \Delta t)$ and synaptic strength change can be computed as.

$$\Delta w(\Delta T) = \int f(v_{MR}(t, \Delta T))\,dt = \xi(\Delta T) \tag{4}$$

which has been shown to result in the same shape illustrated in **Figure 2B** (Zamarreño-Ramos et al., 2011). Furthermore, by reshaping the spike waveform, one can fine tune or completely alter the STDP learning function $\xi(\Delta T)$, as illustrated in **Figure 3**. This way, by building neurons with a given degree of shape programmability, it is possible to change the STDP learning function at will, depending on the application, or make it evolve in time as learning progresses.

Figure 4A shows a way of interconnecting memristors and CMOS neurons for STDP learning. Triangles represent the neuron soma, the flat side indicating its input (dendrites) and the sharp side its output (axon). Dark rectangles are memristors, each representing one synaptic junction. Every neuron controls the voltage at its input (V_{post} in **Figure 4B**) and output (V_{pre} in **Figure 4B**) nodes. When the neuron is not spiking it forces a constant voltage at both nodes, while collecting through its input node the sum of input synaptic spike currents coming from the memristors, which contribute to changing the neuron internal state. When the neuron spikes, it sets a one-spike waveform at both input and output nodes. This way, they send their output spikes forward as pre-synaptic spikes for the destination synaptic memristors, but also backward to preceding synaptic memristors as post-synaptic spikes. Zamarreño et al. showed extensive simulations on these concepts, and how one can change from STDP to anti-STDP by switching polarities of spikes or memristors (Zamarreño-Ramos et al., 2011). For example, (**Figures 3F1,F2**) illustrate the case where forward and backward spikes have opposite polarities, resulting in a symmetric STDP update function $\xi(\Delta T)$. **Figures 3G1,G2** illustrate an example where forward and backward spikes are different, with the backward spike such that its positive part exceeds the positive memristor threshold ($v_{th} = 1.0$). This produces LTD (long term depression) or negative STDP update whenever there is a post-synaptic spike sufficiently apart from a pre-synaptic one; and produces LTP (long term potentiation) if pre- and post-synaptic spikes happen within a given time window (Bichler et al., 2012a,b). **Figures 2H1,H2** illustrate a similar STDP update behavior, except that the update (whether positive or negative) is restricted to a limited time window.

If the system is structured into neural layers (for example, **Figure 4A** shows a 3-neuron-layer system) with memristive synapses in between, then for each layer all pre-synaptic neurons should have the same forward spike shape and all post-synaptic neurons should have the same backward shape. This way, all memristive synapses between these two neural layers will have the same STDP function $\xi(\Delta T)$.

WAVEFORM-DEFINED PLASTICITY IN FERROELECTRIC RESISTIVE SWITCHING MEMRISTORS

In this section, we concentrate on an analysis of resistive switching BiFeO3 (BFO). Our BFO memristors are grown by pulsed laser deposition on Pt/Ti/SiO2/Si substrate with a circular Au top contact (Shuai et al., 2013), see **Figure 5A**. The BFO films have a thickness of some 100 nm. The top contact forms a Schottky diode, causing the created devices to show resistive switching with a rectifying behavior (Shuai et al., 2011). The devices exhibit a combination of voltage- and charge-driven behavior, and are consistent with the requirements of Section Theoretical Principles. When stepping DC voltages across the device, the resistance will follow an exponential curve (Mayr et al., 2012). The voltage level defines the converged resistance value, while the charge passed through the device defines the time frame until this converged value is achieved.

Resistive switching in BFO shows a number of characteristics which make it well-suited for use as a synapse. For instance, the dependence between voltage level and converged resistance makes the BFO devices conform closely to the ideal waveform-driven plasticity postulated in **Figure 3**, as plastic changes in the memristor closely follow the overlapping pre- and post-synaptic waveforms. Up to 8 bit analog resolution can be reliably programmed in the device (Shuai et al., 2013). Due to the Schottky diode, there is also high-ohmic region up to 1 V. Similar to the paradigm of Linn et al. (2010), this can be used in an array of BFO devices to define a voltage readout-region where only a single device in the array is active, eliminating the multiple sneak

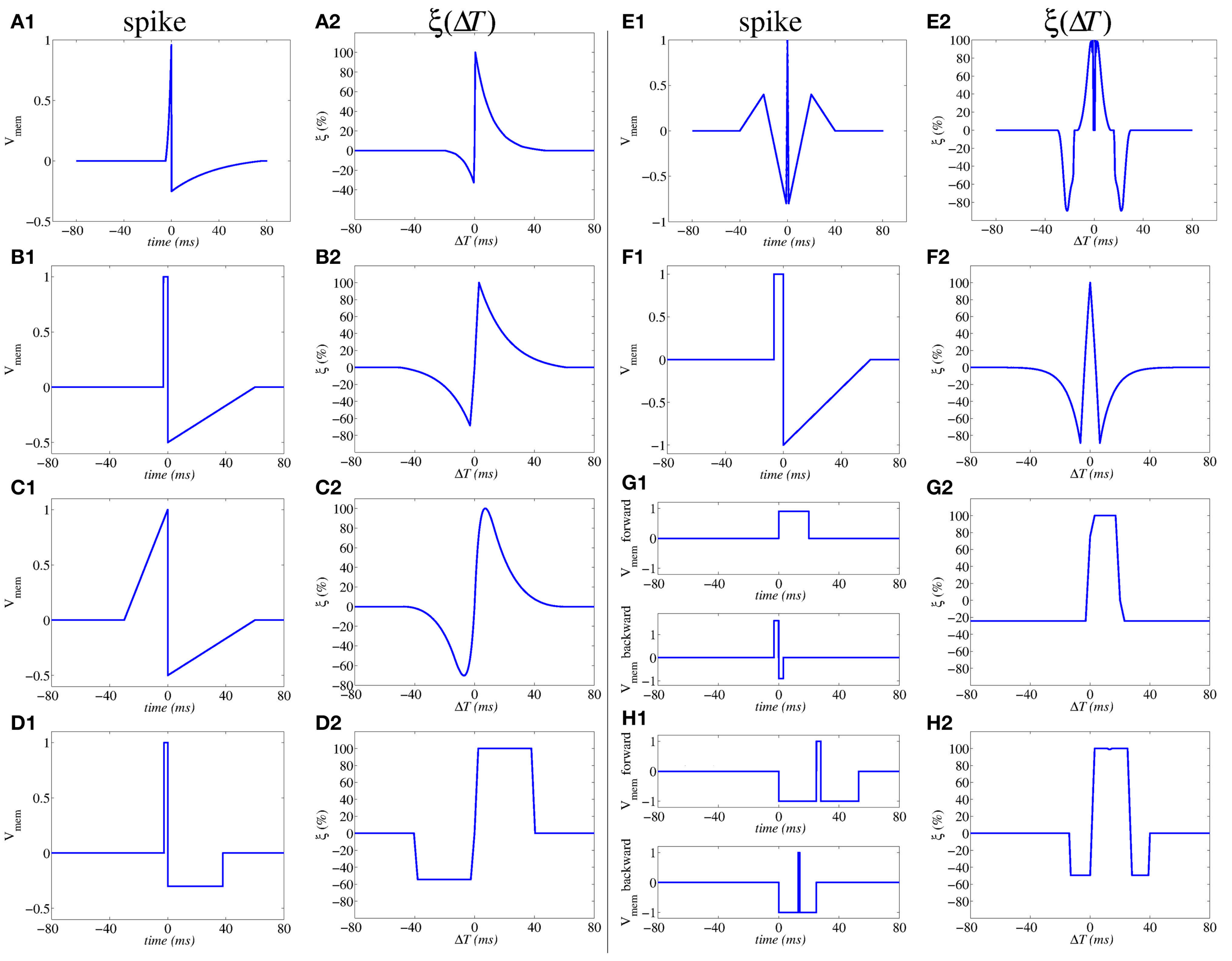


FIGURE 3 | Illustration of influence of action potential shapes on the resulting STDP memristor weight update function $\xi(\Delta T)$. Memristor upper and lower thresholds are normalized to amplitudes ±1.0. From **(A1,A2)** to **(E1,E2)** the same spike waveform travels forward and backward. In **(F1,F2)** the forward and backward waveforms are the same but have opposite polarity. In **(G1,G2)** to **(H1,H2)** the forward and backward waveforms are different. In **(G1,G2)**, the positive pulse of the backward waveform exceeds amplitude +1.0, thus producing negative STDP update whenever there is a post-synaptic spike alone **(G2)**; otherwise if pre- and post-synaptic spikes happen within a given time window, there will be positive STDP update.

current paths that would otherwise severely limit practical array size (Flocke and Noll, 2007). While this characteristic potentially enables large crossbar arrays of BFO devices, defect density is on the order of 30% for an "open circuit" type failure, so a placement algorithm (Mayr et al., 2007) would have to be used in a memristive array to map around defect memristors.

The devices also experience a modification threshold at ca. 2 V, i.e., starting from the Schottky diode threshold at 1 V up to 2 V, the memristance can be measured by the current flow, but the charge inherent in this current does not change the memristance. If appropriate waveforms are chosen, the 2 V threshold extracts pre- and post-synaptic activity correlation as memristance change, as postulated in Section Theoretical Principles. All these voltages are broadly compatible with CMOS logic processes, in contrast to other material choices that need significantly higher voltages (Kuzum et al., 2013).

The waveforms in the upper two curves of **Figure 5B** are used as pre- respectively post-synaptic voltage. Those curves have not been shown in **Figure 3**; however their asymmetry is in the spirit of **Figures 3G1,H1**. These waveforms implement the plasticity model of Mayr et al. (2010), which allows for both rate- and spike-based plastic behavior. In the third curve of **Figure 5B**, which shows the resulting differential voltage across the memristor, the modification thresholds at about 2 V are marked. As can be seen, these are crucial in permitting modification only for true pre-post coincidences (such as at 30 ms), filtering out single pre- or post-synaptic events (such as at 20 ms). The resulting synaptic modification is shown in the last curve of **Figure 5B**, exhibiting a close match with the theoretical model (Mayr et al., 2010).

Measured STDP curves using this paradigm are shown in **Figure 5C**. With their exact reproduction of the waveform-defined exponential time window, they showcase the capability

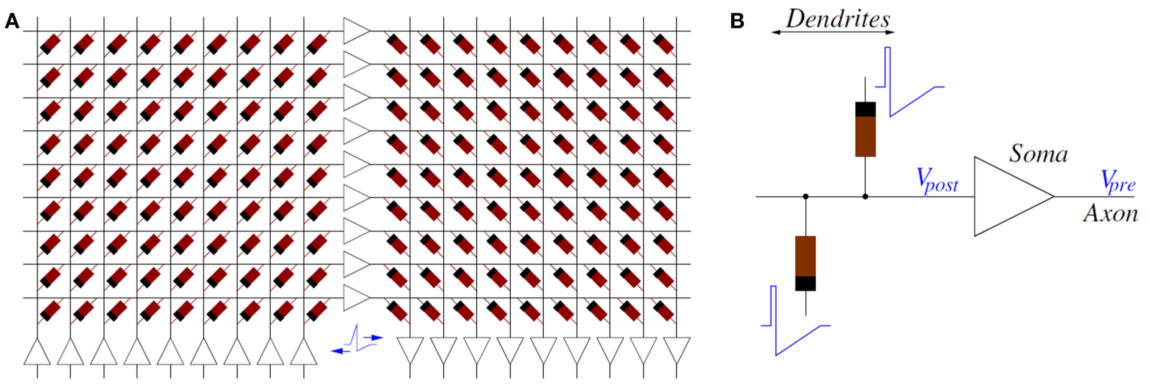


FIGURE 4 | (A) Example of Memristors and CMOS neuron circuits arrangement for achieving STDP learning: feed-forward neural system with 3 layers of neurons and two fully connecting synapse crossbars. **(B)** Details of parts around one post-synaptic neuron. While a neuron is silent, it sets a constant DC voltage at its input (V_{post}) and output (V_{pre}) nodes. When a neuron is sending a spike, it sets a voltage spike at both nodes.

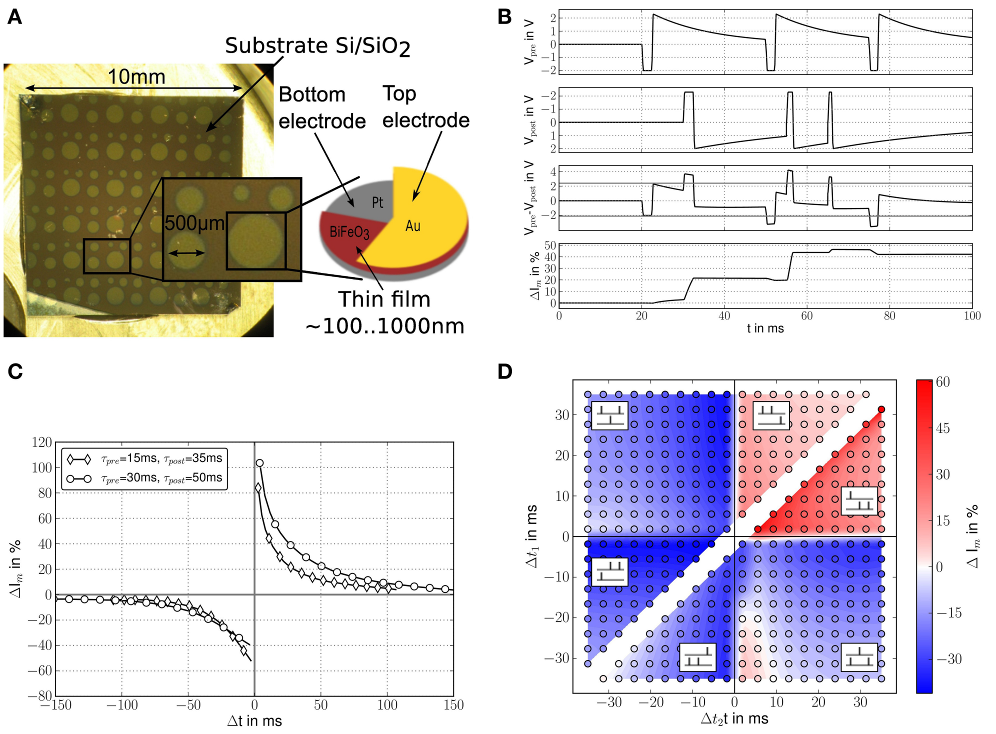


FIGURE 5 | (A) Layout/processing of BiFeO3 devices used (Shuai et al., 2013); **(B)** driving voltage waveforms (from top to bottom): pre-synaptic waveform, post-synaptic waveform, resulting differential voltage across memristor and resulting memristance change shown as percentage change in current through the memristor for a fixed 2 V measurement voltage (Cederstroem et al., 2013); **(C)** measured STDP curves for two different STDP time window settings; time windows are adjusted via the time constants of the exponentials slopes of pre- and post-synaptic waveform, which changes the LTP respectively the LTD part of the STDP window; Weight change as change in current through the memristor; **(D)** measured spike triplet curves (Froemke and Dan, 2002), weight change as change in current through the memristor (Mayr et al., 2012).

of BFO synapses for fine-grained analog weights. In most current memristive materials, the STDP curves deviate significantly more, and their time windows are primarily defined by the physical device characteristics, not the driving waveform (Alibart et al., 2012; Kuzum et al., 2013). In contrast, the voltage-memristance relationship of the BFO synapses lets them conform nicely to the waveform-defines-plasticity paradigm postulated in theory (Zamarreño-Ramos et al., 2011). Through this direct translation of the driving voltage waveforms into the plasticity shape, different time windows can be easily configured via the pre- and post-synaptic waveforms, as can be seen from the two sample curves in **Figure 5C**.

By introducing adaptation into the post-synaptic waveform, specifically an exponential dependence of the post-synaptic action potential duration on the inter-spike interval, the plasticity rule of Mayr and Partzsch (2010) is also able to reproduce triplet and rate plasticity (Froemke and Dan, 2002). When exploring the triplet paradigm with memristors, a faithful reproduction of biological triplet data can be seen (**Figure 5D**), due again to the excellent correlation between driving waveform and evoked memristive plasticity. The post-synaptic adaptation introduced for triplet plasticity can be observed in the different pulse widths in the second curve in **Figure 5B** (Noack et al., 2010).

Defining the plasticity entirely through the waveform can also be used to substantially speed up synapse behavior in BFO up to a 50 μs time scale (You et al., 2014). A switched capacitor system such as (Mayr et al., 2014b), if equipped with a scalable time base (Eisenreich et al., 2009), also offers the intriguing possibility of operating a high-density, CMOS-memristor hybrid neuromorphic system at varying timescales to accommodate different tasks, such as real-time interoperation with a visual sensor vs. offline, high-speed classification tasks where an accelerated timescale leads to faster classification.

HIGH-SPEED PLASTICITY IN FERROELECTRIC TUNNEL MEMRISTORS

"Purely electronic" memristors are nanodevices in which the resistance changes are obtained through electron mediated phenomena at interfaces. These memristors promise an increased endurance and reliability, since the material structure is preserved, as well as a faster switching speed.

The "ferroelectric tunnel memristor" (Bibes et al., 2010) is based on an emerging digital memory concept, subject of intense academic and industrial developments, the ferroelectric resistive RAM (International Technology Roadmap For Semiconductors, 2011). Its base is the ferroelectric tunnel junction (FTJ): an insulating ultrathin (several nanometers) ferroelectric barrier sandwiched between two metallic electrodes (**Figure 6A**). Strain from the substrate assures that the ferroelectric polarization points to one of the electrodes. The polarization can be switched upon application of short voltage pulses and results in resistance changes of up to several orders of magnitude (Garcia et al., 2009; Chanthbouala et al., 2012a). This resistance contrast is linked to different polarization screening in the electrodes: the effective tunneling barrier height dependents on the direction of the ferroelectric layer's polarization and therefore strongly influences the tunneling current. Additionally, the strong non-linearity of the ferroelectric tunnel junction allows for a non-destructive resistance reading at low DC voltage.

By designing the devices in such way that the switching occurs through non-uniform ferroelectric domain configurations, quasi-analog resistance variations can be obtained (Chanthbouala et al., 2012b). A direct link between these intermediate resistance states and the ferroelectric domain configuration allows the description of its dynamic behavior through models of domain nucleation and growth in ferroelectric films. Furthermore, the cumulative behavior upon application of trains of voltage pulses has already been demonstrated. As the polarization reversal process in the ferroelectric film depends on pulse amplitude and duration, these parameters can be adapted to achieve the desired resistance change in the memristive device—a very promising feature for the implementation of STDP-based learning with ferroelectric tunnel memristors (Chanthbouala et al., 2012b).

It has recently been demonstrated that fully-patterned solid-state ferroelectric tunnel memristors based on $BiFeO_3$ (fully patterned submicron $Co/BiFeO_3/Ca_{0.96}Ce_{0.04}MnO_3$ tunnel junctions) can be produced with high yield and with low device-to-device variations. They show resistance contrasts of more than 3 orders of magnitude, can be commuted with pulses of 100 ns and amplitudes of about 2 V, and have a large endurance of over 4×10^6 cycles (Boyn et al., 2014).

In **Figure 6**, we plot as in Yamada et al. (2013) the multilevel behavior of a ferroelectric tunnel memristor depending on applied voltages. The curves in **Figure 6B** show the DC resistance value of the device after writing pulses of different amplitudes. To use this memristor as a plastic synapse we consider $-V_{MR}$ to represent the time difference $\Delta T = t_{post} - t_{pre}$. Then $\Delta T > 0$, i.e., $V_{MR} < 0$ in **Figure 6B**, implies increasing conductance that corresponds to Hebb's rule. Conversely, $\Delta T < 0$ results in a decrease of the synaptic weight.

Choosing the waveform of **Figure 3B1** for pre- and post-synaptic voltage neurons, the width of the positive square pulse can be as low as 100 ns in the case of the ferroelectric tunnel memristor. Accordingly, the ramp phase of the waveform will be a few times larger than this. As a result, the time difference between spikes for the STDP shown in **Figure 3B2** can be less than 1 μs.

SPIN-TRANSFER TORQUE MAGNETIC TUNNEL JUNCTION AS A STOCHASTIC SYNAPSE

Spin-Transfer Torque Magnetic Tunnel Junctions (STT-MTJs) constitute another choice to implement plastic non-volatile synapses. They rely on a different operating mechanism than the devices presented in the rest of the paper, and for this reason are not always thought as memristive devices. Their specific stochastic behavior, however, can be particularly interesting for synaptic applications. And as they constitute the basic cell of the second generation of Spin Transfer Torque Magnetic RAM (STT-MRAM)—which is currently reaching the market—, they present a high level of CMOS compatibility and of maturity.

The basic structure of a STT-MTJ is presented in **Figure 7A** and is constituted by an ensemble of layers of different materials. The magnetic "fixed" layer is a small magnet whose magnetization is pinned in one direction. The magnetic "free" layer is a thinner magnet whose magnetization can be either parallel (P) or antiparallel (AP) to the one of the fixed layer. Due to the Tunnel Magnetoresistance effect, the electrical resistance of the P and AP state is different. And due to the Spin Transfer Torque effect, a positive current can switch the device from AP to P state, and a negative current can switch the device from P to AP state. This leads to the I–V curve seen in **Figure 7B**, which is reminiscent of a memristive device. However, MTJs are truly binary device: AP and P states are the only possible states. Some proposals exist to increase the number of states (Lou et al., 2008) or to include another physical effect (domain wall motion) in the MTJ to reach multilevel behavior (Wang et al., 2009; Chanthbouala et al., 2011). However, these variations do not exhibit the same degree of maturity as binary STT-MTJs. In comparison with traditional memristive devices, STT-MTJs are fast to write (programming can be as fast as 1.5 ns) and possess outstanding endurance (switching the free layer magnet is not associating with an aging mechanism). Their main drawback is a relatively high fabrication cost and a low R_{OFF}/R_{ON} ratio. STT-MTJs should be associated with different CMOS circuits than other memristive devices for this reason (Zhang et al., 2014).

Additionally, a specificity of STT-MTJs, of special interest for synaptic applications, is that switching is stochastic. When one applies a programming pulse, a STT-MTJ has only a *probability* to switch state, which is independent of the STT-MTJ's history: every time the programming pulse is applied, the STT-MTJ has the same probability to switch state. This is well-seen on the experimental measurements of **Figure 7C** on devices of Devolder et al. (2008), Marins de Castro et al. (2012). The switching probability can be controlled by programming voltage and pulse duration. The basic physics behind this effect is well-understood (Diao et al., 2007; Devolder et al., 2008) and we have recently developed a comprehensive analytical model of it for circuits and systems designers (Vincent et al., 2015). As seen in **Figure 7D**, a striking feature is that the mean switching time of the STT-MTJ can be adjusted over many orders of magnitude by choosing the programming current. It has also been proven that STT-MTJ stochastic switching can be used to generate high quality random numbers that pass standardized statistical tests qualifying true random number generators (Fukushima et al., 2014). Stochastic switching can also be adjusted by layout of the junctions (size and eccentricity).

STT-MTJs are suitable for implementing a stochastic version of STDP that has been studied in several recent works (Kavehei, 2013; Suri et al., 2013; Yu et al., 2013; Vincent et al., 2014). They exploit, at the system level, a functional equivalence (Goldberg et al., 2001) that exists between multi-level deterministic synapses and binary probabilistic synapses. When a long term potentiation or depression occurs, instead of changing the conductance of the synapse partially, stochastic STDP has a small probability of changing it totally. And if several STT-MTJs are connected in parallel, a multibit synapse can be emulated. Since STT-MTJs have no internal dynamic besides stochastic switching, stochastic STDP can be implemented using similar strategies to the one used for ferroelectric devices. Only the behavior at the system level will be different.

In our works, we have been working with a stochastic version of the simplified version of STDP which is theorized in Nessler et al. (2013) and also used in Bichler et al. (2012a), Suri et al. (2013), Querlioz et al. (2013), and similar to the one of **Figures 3G1,G2**. A possible implementation with STT-MTJs is summarized on **Figure 7E**. It relies on overlapping pulses, but with clear separation of transmission and programming operation (Suri et al., 2013; Vincent et al., 2014). Although very simple, this STDP rule can lead to complex machine learning tasks like learning to detect cars on a video (Vincent et al., 2014). Additionally, we have observed that it is surprisingly robust to STT-MTJ variability (Vincent et al., 2014). However, this is just an example and other forms of STDP may be implemented with STT-MTJs if one accepts their stochastic nature.

SRDP WITH MEMRISTIVE DEVICES

The learning process described in the previous section has been implemented in a large variety of solid state memory devices with non-volatile characteristics. However, if we consider the synaptic plasticity mechanisms observed in biological computing systems, modification of the synaptic efficiency (evaluated by measuring the transmission of a single spike and equivalent to the synaptic weight) can be either permanent (i.e., lasting for months to years) or temporary (i.e., relaxing to its initial state with a characteristic time constant in the milliseconds to hours range). This observation leads to the definition of Long Term Plasticity

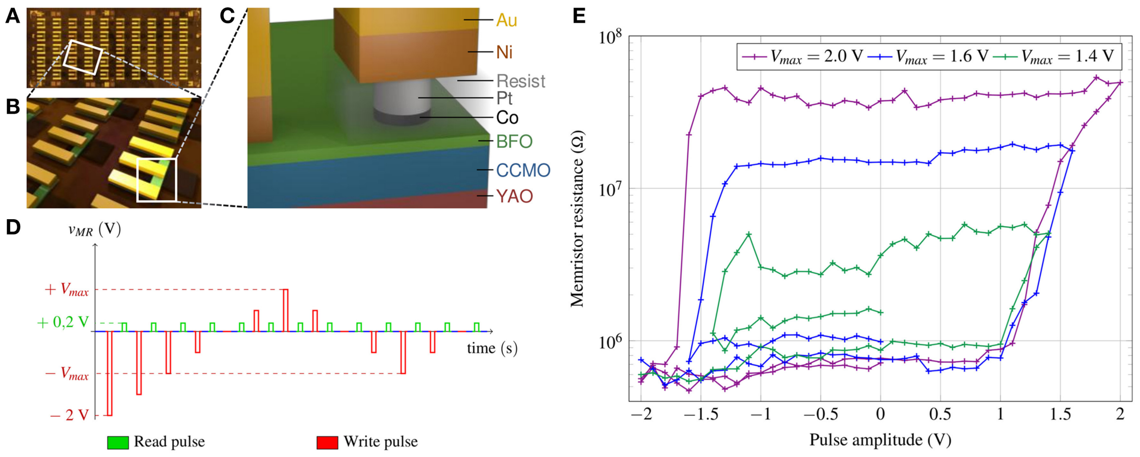


FIGURE 6 | (A) Optical microscope image of the chip after patterning showing 5×10 ferroelectric tunnel junctions (FTJ); **(B)** 3D representation of a zoomed area containing a few FTJs. The three parallel bars are the ground-signal-ground contact pads; **(C)** 3D sketch of one FTJ (Boyn et al., 2014); **(D)** schema of the voltages applied to the memristor. The reading pulse V_{read} is lower than the threshold ($V_{read} = 200$ mV). Writing is performed by the application of 100 ns voltage pulses of different amplitudes. The writing voltages increase from −2 V to V_{max} by a step of 0.1 V. Then, the amplitude of the writing pulses decreases to $-V_{max}$; **(E)** dependence of the resistance of the ferroelectric tunnel memristor measured at V_{read} on the applied writing cycles. The different curves correspond to different consecutive measurements with varying V_{max}.

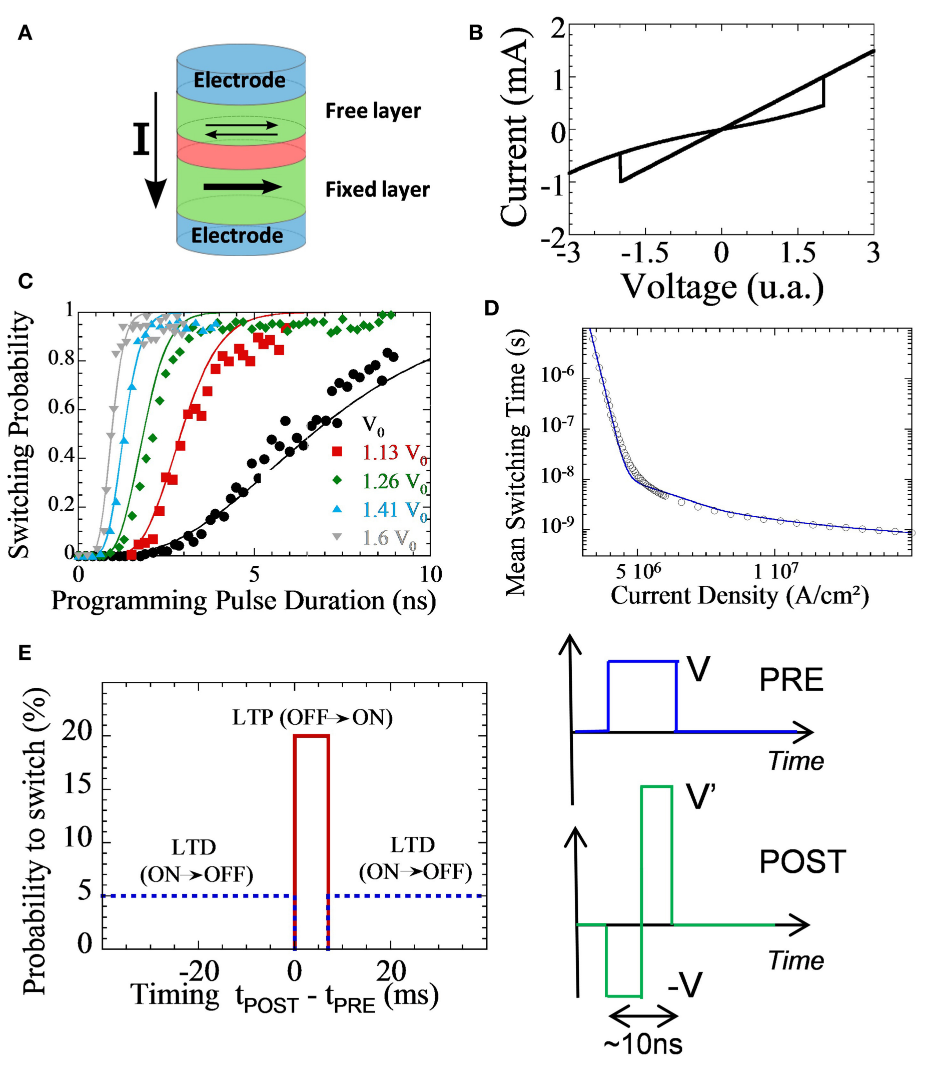


FIGURE 7 | From Vincent et al. (2014, 2015). (A) Cartoon of a Spin Transfer Torque Magnetic Tunnel Junction (STT-MTJ). **(B)** Typical I–V curve of the STT-MTJ. **(C)** Experimental measurements of stochastic switching. **(D)** Model of mean switching time as a function of programming current. **(E)** Our simplified STDP rule and PRE and POST overlapping pulses which implement it naturally with STT-MTJs.

(LTP) and Short Term Plasticity (STP), respectively. We can notice that the boundary classification into Long Term (LT) and Short Term (ST) effects is not well-defined and should be considered with respect to the required task. Both STP and LTP can correspond to an increase or decrease of the synaptic efficiency thus leading to the definition of Short Term (Long Term) potentiation and depression, respectively. In biology, synaptic plasticity can be attributed to various mechanisms involved in the transmission of the signal between a pre- and post-neuron, such as neurotransmitter release modification, neurotransmitter recovery in the pre-synaptic connection, receptors sensitivity modification or even structural modification of the synaptic connection (see Bliss and Collingridge, 1993), for a description of the different mechanisms involved in STP and LTP). Based on this observation, two important points need to be stressed. First, STP and LTP processes are not restricted to a particular learning strategy (i.e., STDP and SRDP, for example). In this section, we present examples of STP and LTP processes based on a particular case of rate coding strategy but these considerations are still valid for other coding strategies (see Alibart et al., 2012, for STDP with STP devices). Secondly, if plasticity is intimately linked to the notion of learning, it is important to notice that there is no one-to-one equivalence between the concepts of STP, LTP and the notion of Short Term Memory (STM) and Long Term Memory (LTM). Indeed, even if a direct parallel has been proposed based on the particular concept of memory consolidation (Lamprecht and Ledoux, 2004), which corresponds to accumulation of Short Term effect leading to Long Term memory, there are still very important questions to be answered about how learning (and the associated synaptic plasticity) is related to the memorization of information that can also present different time scale from milliseconds to years.

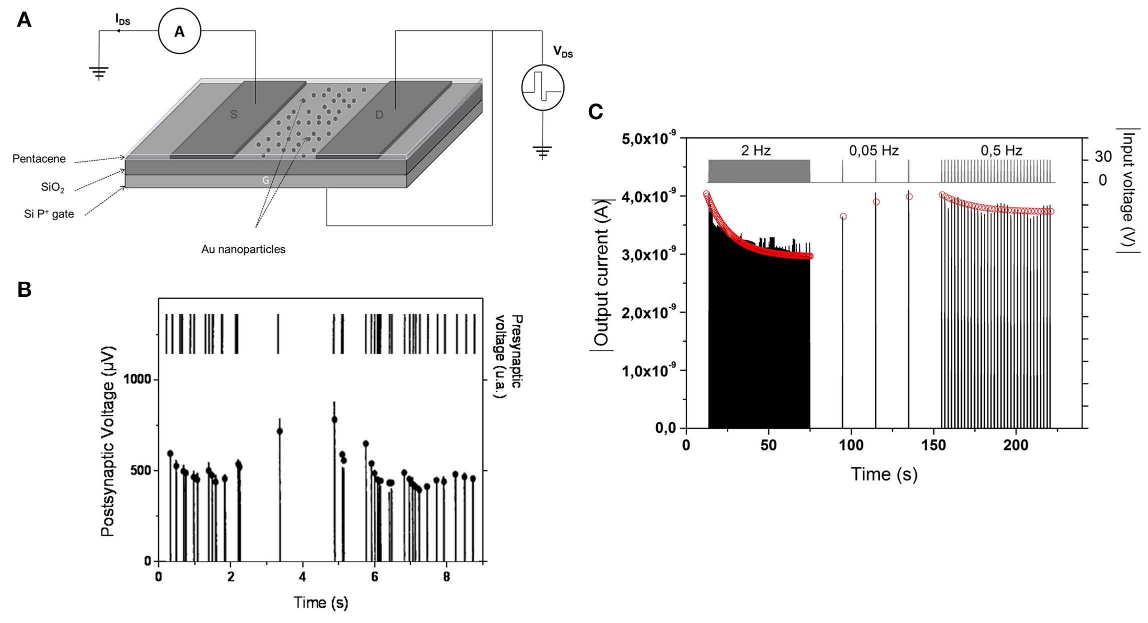


FIGURE 8 | (A) Schematic of the NOMFET in a diode-like configuration. This leaky memory transistor was proposed to implement the Short Term Plasticity **(B)** STP measured in biological synapses (Varela et al., 1997) and **(C)** STP implement in solid state device.

SHORT TERM PLASTICITY (STP)

Implementation of STP has been proposed in a variety of nanoscale memory devices. The first proposition of STP was realized in a nanoparticles/organic memory transistor (NOMFET)—**Figure 8** (Alibart et al., 2010). The basic principle of this device is equivalent to a floating gate transistor. Charges are stored in the nanoparticles and modify the channel conductivity via Coulombic repulsion between the carriers (holes) and the charged nanoparticles. The particularity of this device is to present a leaky memory behavior: charges stored in the nanoparticles tend to relax with a characteristic time constant in the 1–100 s range. When the NOMFET is connected in a diode like configuration (**Figure 8A**), each input spike (with a negative voltage value) charges the nanoparticles and decreases the NOMFET conductivity. Between pulses, charges escape from the nanoparticles and the conductivity relaxes toward its resting value. By analogy with biology, this device mimics the STP observed in depressing synapses (**Figures 8B,C**) and described by Abbott et al. (1997). As a matter of comparison, this synaptic functionality is realized with a single memory transistor when its implementation in Si based technologies (i.e., CMOS) required 7 transistors (Boegerhausen et al., 2003). STP has been also demonstrated in two-terminal devices that would ensure higher device density when integrated into complex systems. Equivalently, STP in two terminals devices is implemented by taking advantage of the volatility of the different memory technologies (i.e., low retention of the state that is often a drawback in pure memory applications). Cationic redox systems based on Electro-Chemical Memory cells (ECM) (Ohno et al., 2011) or anionic Valence Change Memory (Chang et al., 2011; Yang et al., 2013) have demonstrated STP with a facilitating behavior. In such devices, Short Term Plasticity is ensured by the low stability of the conducting filaments that tend to dissolve, thus relaxing the device toward the insulating state. TiO2 VCM cells have been reported with both facilitating and depressing behavior (Lim et al., 2013) with relaxation related to oxidation-reduction counter reaction. Protonic devices have demonstrated STP with depressing functionality due to proton recovery latency from atmosphere required to restore the proton concentration and conductivity (Josberger et al., 2014).

In the case of rate dependent plasticity, STP can be of depressing type (i.e., decrease of the synaptic efficiency when synaptic activity increase) or facilitating type (i.e., increase of synaptic efficiency when synaptic activity increase). In terms of functionality, Abbott et al. (1997) has demonstrated that depressing synapses with STP act as a gain control device (at high frequency, i.e., high synaptic activity, the synaptic weight is decreased, thus leading to a reduction of the signal when activity becomes too important). More generally, STP (both depressing and facilitating) provides a very important frequency coding property (as depicted in **Figures 8B,C**) that could play a major role in the processing of spike-rate coded information. Indeed, if a simple Integrate and Fire neuron (I&F) is associated with static weight (with no dependence with spike frequency), the computing node (i.e., neuron and synapses) is only a linear filter (linear combination of the different input) while STP turns the node to non-linear. This property can be used to implement reservoir computing approaches as proposed by Maass (Buonomano and Maass, 2009) with the Liquid State Machine and could be an important property of biological systems for computation.

CO-EXISTENCE OF STP AND LTP IN THE SAME DEVICE

If the contribution of ST and LT processes to computing is not completely understood in biological systems, we should consider that both STP and LTP effects in synaptic connections are required in neuro-inspired computing systems. A first approach is to consider that repetition of short term effects should lead to Long Term modification in the synaptic connections. This behavior would explain the important hypothesis of memory consolidation in the sense of psychology (Lamprecht and Ledoux, 2004). Ohno et al. (2011) reported the coexistence of Long Term and Short Term Potentiation in atomic bridge technology (**Figure 9**). Depending on pre-synaptic activity (associated to spike rate in this case), the synaptic conductivity is increased due to the formation of a Ag filament across the insulating gap. While for low frequency, the bridge tends to relax between pulses, higher frequencies lead to a strong filament that maintains the device in the ON state. These results suggest a critical size of the bridging filament in order to maintain the conductive state (i.e., providing a LTP of the synaptic connection). Similar results have been obtained in a variety of memory devices where filamentary switching displayed two regimes of volatility. Chang et al. (2011) have evidenced a continuous evolution of the volatility as a function of the conductivity level of the device in WO3 oxide cells attributed to the competition between oxygen vacancies drift (creation of conductive path across the device) and lateral diffusion (disruption of the conducting filaments). Another description of these two regimes of volatility could be associated to a competition between surface and volume energies in the conductive filament.

If this transition between Short Term Plasticity and Long Term Plasticity is intuitively well-associated to the concept of STM to LTM learning in psychology, we can note that it induces some restriction in term of network functionality. Indeed, in biology, the facilitating process observed at short time scale and associated to an increase of neurotransmitter release probability during a burst of spike (i.e., corresponding to an increase of synaptic efficiency at high frequency spiking rate) is additive with LTP (Bliss and Collingridge, 1993). In this case the node (neuron and synapses) maintains its rate coding property (associated to short term process and described previously as a non-linear node) and can also display long term modification of the synaptic weight. Alternative approaches are still needed as proposed by Cantley et al. (2011) where Short Term processes and Long Term Processes are realized by two different devices (leaky floating gate transistor and non-volatile two-terminal devices) in order to match the complexity of biological synapses. One fundamental issue that needs to be explored is the balance between the device functionality required for proper operation of computing systems (i.e., performances) and optimal integration in order to match synaptic density required for computing.

TOWARD MEMRISTOR-CMOS ARCHITECTURES AND CIRCUITS

In order to exploit the plasticity of memristor-based artificial synapses, specific circuit architecture needs to be developed. Indeed, depending on the polarity and electrical characteristics of investigated devices, two types of circuits have been identified which are described in the following paragraphs.

CIRCUITS FOR BIPOLAR MEMRISTORS

Most of the works on memristive devices that have been published over the last couple of years focus on bipolar resistive switching devices (Waser and Aono, 2007; Snider, 2008; Strukov et al., 2008; Jo et al., 2010). This is the case for all the devices presented in Section STDP Learning Thanks to Overlapping Events. These devices exhibit characteristics close to the original Memristor predicted by Chua. Their resistance can be increased or decreased with opposite polarity voltage pulses and the resistance change is cumulative with the previous state of the device, which makes them particularly suitable to implement synaptic-like functionality.

A biologically-inspired spiking NN-based computing paradigm which exploits the specific physics of those devices is presented in Querlioz et al. (2011, 2013). In this approach, CMOS input and output neurons are connected by bipolar memristive devices used as synapses. It is natural to lay out the nanodevices in the widely studied crossbar as illustrated on **Figure 10**. Learning is competitive thanks to lateral inhibition and fully unsupervised using a simplified form of STDP.

Using this topology, performance comparable to traditional supervised networks has been measured (Querlioz et al., 2013) for the textbook case of character recognition, despite extreme variations of various memristive device parameters. With the same approach, unsupervised learning of temporally correlated patterns from a spiking silicon retina has also been demonstrated. When tested with real-life data, the system is able to extract complex and overlapping temporally correlated features such as car trajectories on a freeway (Bichler et al., 2012a).

CIRCUITS FOR UNIPOLAR MEMRISTORS

All that we have discussed in this work can be adapted to another class of memristive devices—the unipolar devices where all applied voltages to increase or decrease the resistance value are positive. Among them, in particular, Phase-Change Memory (PCM) has good maturity, scaling capability, high endurance, and good reliability (Fantini et al., 2010). PCM resistance can be modified by applying a temporal temperature gradient modifying the material organization between an amorphous and a crystalline phase. The amorphous region inside the phase change layer can be crystallized by applying set pulses, thus increasing device conductance. It was shown that the magnitude of the relative increase in conductance can be controlled by the pulse amplitude and by the equivalent pulse width (Kuzum et al., 2012). Amorphization, on the other hand, is a more power-hungry process and is not progressive with identical pulses. The current required for amorphization is typically 5–10 times higher than for crystallization, even for state-of-the art devices.

To overcome these issues, a novel low-power architecture "2-PCM Synapse" was introduced in Bichler et al. (2012b). The idea is to emulate synaptic functions in large scale neural networks using two PCM devices constituting one synapse as shown in **Figure 11**. These two devices have an opposite contribution to the neuron's integration. When the synapse needs to be potentiated, the Long Term Potentiation (LTP) PCM device undergoes a partial crystallization, increasing the equivalent weight of the synapse. Similarly, when the synapse must be depressed, the Long

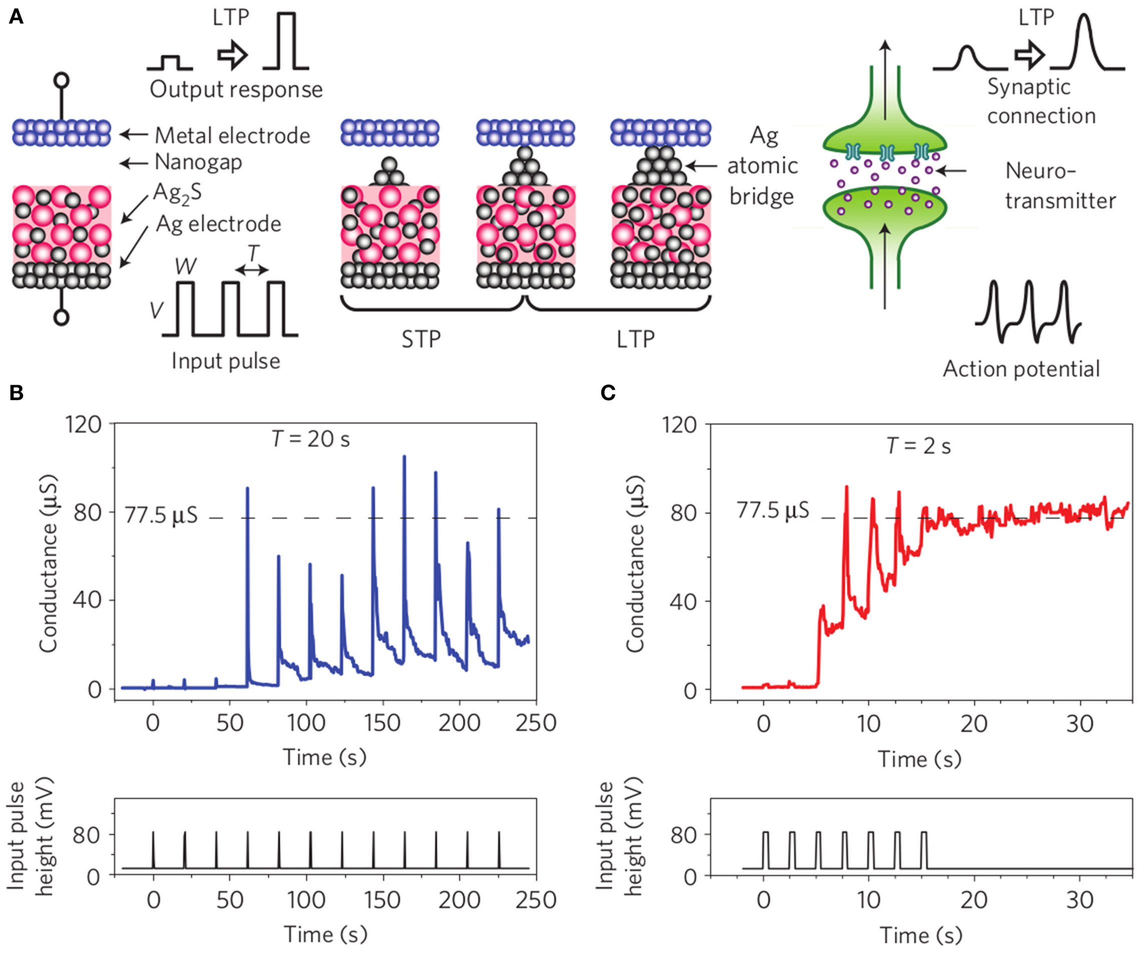


FIGURE 9 | Adapted from Ohno et al. (2011). (A) Schematic of atomic bridge devices that was proposed for Short Term Plasticity, Long Term Plasticty (STP/LTP) transition demonstration. Depending on the spiking activity, **(B)** the metallic filament do not bridge the two electrodes and tends to relax toward the OFF state while it remains **(C)** in the ON state once it bridges the two electrodes.

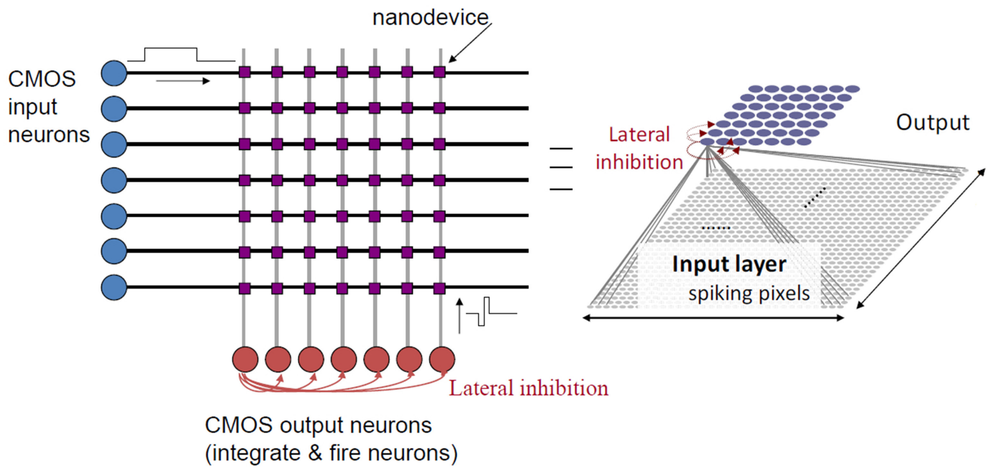


FIGURE 10 | Basic crossbar circuit topology. Wires originate from CMOS input layer (horizontal black wires) and from the CMOS output layer (vertical gray wires). Memristive nanodevices are located at the cross points of the horizontal and vertical wires.

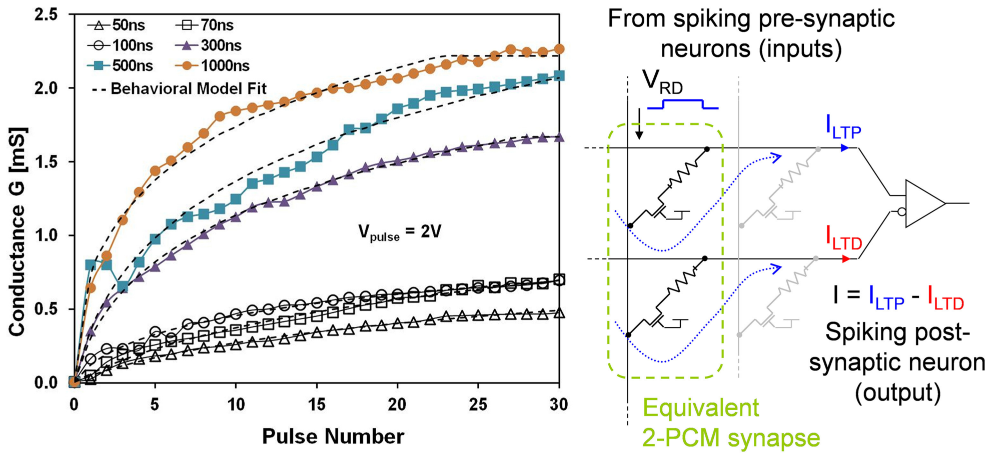


FIGURE 11 | Left: Experimental LTP characteristics of the unipolar PCM device. **Right:** Principle of an equivalent bipolar synapse realized with a 2-PCM circuit. Note that the neuron circuit is not represented on the schematic.

Term Depression (LTD) PCM device is crystallized. As the LTD device has a negative contribution to the neuron's integration, the equivalent weight of the synapse is reduced. Furthermore, because gradual crystallization is achieved with successive identical voltage pulses, the pulse generation is greatly simplified. Note however that such synaptic circuit will require a slightly more complex post-synaptic neuron circuit in order to deal with pulse integration and generation. This should have a limited impact on the overall neuromorphic circuit given the lower number of neurons vs. synapses.

DISCUSSION

Memristive devices are an appealing solution to implement plastic synapses, if we develop the specific driving signals to emulate different learning rules. The most popular synaptic plasticity implementation is based on the realization of Hebbian learning, and in particular of STDP. We shall however note that other plasticity mechanisms exist that have been studied and modeled as suggested in a recent work (Kornijcuk et al., 2014). In this paper, we focused on different implementations of STDP, by taking advantage of the device physics of different memristive devices. The functional differences in the behaviors of the devices directly translate into differences in the learning rules (real time or accelerated, deterministic or stochastic). Using other devices, we also presented other synaptic ideas, such as short term plasticity, or those which exploit interactions between short term and long term plasticity. Finally, we proposed some implementation ideas, offering a large overview of the different possibilities in several material systems.

As memristors are primarily targeted toward future high-density nanoscale arrays, CMOS driver circuits need to be scaled to these dimensions as well. That is to say, the required neuromorphic driver circuits need to be moved to deep submicron technologies. One recently presented method to achieve this is the use of switched-capacitor neuromorphic circuits, which are able to implement the required analog waveforms in high density technologies as small as 28 nm (Mayr et al., 2014b). Coupled with deep submicron CMOS sensors (Henker et al., 2007), they offer the possibility of a full image processing pyramid based on memristive computation in a nanoscale CMOS-memristor hybrid. However, developing appropriate and highly scaled driver circuits for memristive synapses which do not bring large overheads is a significant goal for today's research. This is especially true for proposals that exploit passive crossbar integration. Such circuit topology is particularly appealing for neuromorphic engineers as it offers a direct equivalent for the neuron/synapse circuit with high parallelism and high integration density in which a single device is associated to a single synapse between two neurons (input line and output column). However, it brings circuit challenges (crosstalk, sneak path, impedance mismatch,...) that need to be overcome.

From a more systems' perspective, the most interesting applications for nanoscale memristors will be those that require a large number of learned or programmed synaptic weights. It is important to already consider such applications, to understand the true impact of memristive technology. One of these applications is the Neural Engineering Framework (Eliasmith and Anderson, 2004), which can be used to implement straightforward signal computation, sensor fusion (Mayr et al., 2014a), and recognition (Bichler et al., 2012a), but also models of cognition (Eliasmith et al., 2012). The large number of synapses offered by nanoscale memristive arrays makes the implementation of complex cognitive processing of such large-scale models (Eliasmith et al., 2012) on a single CMOS-memristor hybrid IC a real possibility.

Finally, it is important to understand that there are no absolute optimal memristive devices for the implementation of plasticity in hardware neural networks. The variety of behaviors observed in today's research will be an advantage for neuromorphic chip designers and computational neuroscientists since it opens new paths of implementation of neural computations. In this respect, the plastic behaviors measured on memristive devices and presented in this paper provide the primitive for future neuromorphic breakthroughs.

ACKNOWLEDGMENTS

This paper was partly supported by funding under the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement no. 269459 (CORONET), no. 612058 (RAMP), no. 618024 (BAMBI), no. 318597 (SYMONE), no. 259068 (NANOBRAIN), no. 604102 (HBP), under the French National Research Agency under grant agreement ANR-11-NANO-0002 (MHANN), ANR-13-JS03-0004 (COGNISPIN), ANR-12-PDOC-0027 (DINAMO), ANR-12-BS03-010 (SYNAPTOR), under the Spanish Ministerio de Economía y Competitividad under ERANET grant PRI-PIMCHI-0768 (PNEUMA), under Spanish research grants (with support from the European Regional Development Fund) TEC2012-37868-C04-01 (BIOSENSE), under Andalusian research grant TIC-6091 (NANONEURO).

REFERENCES

Abbott, L. F., Varela, J. A., Sen, K., and Nelson, S. B. (1997). Synaptic depression and cortical gain control. *Science* 275, 221–224. doi: 10.1126/science.275.5297.221

Alibart, F., Pleutin, S., Bichler, O., Gamrat, C., Serrano-Gotarredona, T., Linares-Barranco, B., et al. (2012). A memristive nanoparticle/organic hybrid synapstor for neuroinspired computing. *Adv. Funct. Mater.* 22, 609–616. doi: 10.1002/adfm.201101935

Alibart, F., Pleutin, S., Guérin, D., Novembre, C., Lenfant, S., Lmimouni, K., et al. (2010). An organic nanoparticle transistor behaving as a biological spiking synapse. *Adv. Funct. Mater.* 19, 330–337. doi: 10.1002/adfm.200901335

Baek, I. G., Lee, M. S., Seo, S., Lee, M. J., Seo, D. H., Suh, D.-S., et al. (2004). Highly scalable nonvolatile resistive memory using simple binary oxide driven by asymmetric unipolar voltage pulses. *Tech. Dig. IEEE Int. Electron Devices Meet.* 587–590. doi: 10.1109/IEDM.2004.1419228

Bi, G., and 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., and Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annu. Rev. Neurosci.* 24, 139–166. doi: 10.1146/annurev.neuro.24.1.139

Bibes, M., Grollier, J., Barthélémy, A., and Mage, J.-C. (2010). *Ferroelectric Device with Adjustable Resistance*. Patent WO 2010142762 A1.

Bichler, O., Querlioz, D., Thorpe, S. J., Bourgoin, J.-P., and Gamrat, C. (2012a). Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Netw.* 32, 339–348. doi: 10.1016/j.neunet.2012.02.022

Bichler, O., Suri, M., Querlioz, D., Vuillaume, D., DeSalvo, B., and Gamrat, C. (2012b). Visual pattern extraction using energy-efficient "2-PCM synapse" neuromorphic architecture. *IEEE Trans. Electron Devices* 59, 2206–2214. doi: 10.1109/TED.2012.2197951

Bliss, T. V. P., and Collingridge, G. L. (1993). A synaptic model of memory: long-term potentiation in the hippocampus. *Nature* 361, 31. doi: 10.1038/361031a0

Boegerhausen, M., Suter, P., and Liu, S.-C. (2003). Modeling short-term synaptic depression in silicon. *Neural Comput.* 15, 331–348. doi: 10.1162/089976603762552942

Boyn, S., Girod, S., Garcia, V., Fusil, S., Xavier, S., Deranlot, C., et al. (2014). High-performance ferroelectric memory based on fully patterned tunnel junctions. *Appl. Phys. Lett.* 104, 052909. doi: 10.1063/1.4864100

Buonomano, D. V., and Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nat. Rev. Neurosci.* 10, 113–125. doi: 10.1038/nrn2558

Cantley, K. D., Subramaniam, A., Stiegler, H. J., Chapman, R. A., and Vogel, E. M. (2011). Hebbian learning in spiking neural networks with nanocrystalline silicon TFTs and memristive synapses. *IEEE Trans. Nanotechnol.* 5, 1066. doi: 10.1109/TNANO.2011.2105887

Cassenaer, S., and Laurent, G. (2007). Hebbian STDP in mushroom bodies facilitates the synchronous flow of olfactory information in locusts. *Nature* 448, 709–713. doi: 10.1038/nature05973

Cederstroem, L., Starke, P., Mayr, C., Shuai, Y., Schmidt, H., and Schüffny, R. (2013). A model based comparison of BiFeO3 device applicability in neuromorphic hardware. *IEEE Int. Symp. Circuits Syst.* 2323–2326. doi: 10.1109/ISCAS.2013.6572343

Chang, T., Jo, S.-H., and Lu, W. (2011). Short term memory to long term memory transition in a nanoscale memristor. *ACS Nano*. 5, 7669–7676. doi: 10.1021/nn202983n

Chanthbouala, A., Crassous, A., Garcia, V., Bouzehouane, K., Fusil, S., Moya, X., et al. (2012a). Solid-state memories based on ferroelectric tunnel junctions. *Nature Nano*. 7, 101. doi: 10.1038/nnano.2011.213

Chanthbouala, A., Garcia, V., Cherifi, R. O., Bouzehouane, K., Fusil, S., Moya, X., et al. (2012b). A ferroelectric memristor. *Nature Mater*. 11, 860–864. doi: 10.1038/nmat3415

Chanthbouala, A., Matsumoto, R., Grollier, J., Cros, V., Anane, A., Fert, A., et al. (2011). Vertical-current-induced domain-wall motion in MgO-based magnetic tunnel junctions with low current densities. *Nat. Phys.* 7, 626–630. doi: 10.1038/nphys1968

Chua, L. (1971). Memristor-missing circuit element. *IEEE Trans. Circuit Theor.* 18, 507–519. doi: 10.1109/TCT.1971.1083337

Chua, L. (2014). If it's pinched it's a memristor. *Semicond. Sci. Technol.* 29:104001. doi: 10.1088/0268-1242/29/10/104001

Chua, L. O., and Kang, S. M. (1976). Memristive devices and systems. *Proc. IEEE* 64, 209–223. doi: 10.1109/PROC.1976.10092

Devolder, T., Hayakawa, J., Ito, K., Takahashi, H., Ikeda, S., Crozat, P., et al. (2008). Single-shot time-resolved measurements of nanosecond-scale spin-transfer induced switching: stochastic versus deterministic aspects. *Phys. Rev. Lett.* 100:057206. doi: 10.1103/PhysRevLett.100.057206

Diao, Z., Li, Z., Wang, S., Ding, Y., Panchula, A., Chen, E., et al. (2007). Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory. *J. Phys. Condens. Matter*. 19:165209. doi: 10.1088/0953-8984/19/16/165209

Eisenreich, H., Mayr, C., Henker, S., Wickert, M., and Schüffny, R. (2009). A novel ADPLL design using successive approximation frequency control. *Microelectron. J.* 40, 1613–1622. doi: 10.1016/j.mejo.2008.12.005

Eliasmith, C., and Anderson, C. C. H. (2004). *Neural Engineering: Computation, Representation, and Dynamics. Neurobiological Systems*. Cambridge, MA: MIT Press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., et al. (2012). A large-scale model of the functioning brain. *Science* 338, 1202–1205. doi: 10.1126/science.1225266

Fantini, A., Sousa, V., Perniola, L., Gourvest, E., Bastien, J. C., Maitrejean, S., et al. (2010). "N-doped GeTe as performance booster for embedded phase-change memories," in *International Electron Devices Meeting* (San Francisco, CA), 29.1.1–29.1.4. doi: 10.1109/IEDM.2010.5703441

Feldman, D. (2000). Timing-based LTP and LTD at vertical inputs to layer II/III pyramidal cells in rat barrel cortex. *Neuron* 27, 45–56. doi: 10.1016/S0896-6273(00)00008-8

Fieres, J., Schemmel, J., and Meier, K. (2008). "Realizing biological spiking network models in a configurable wafer-scale hardware system," in *IEEE International Joint Conference Neural Network* (Hong Kong), 969–976. doi: 10.1109/IJCNN.2008.4633916

Finelli, L. A., Haney, S., Bazhenov, M., Stopfer, M., and Sejnowski, T. J. (2008). Synaptic learning rules and sparse coding in a model sensory system. *PLoS Comput. Biol.* 4:e1000062. doi: 10.1371/journal.pcbi.1000062

Flocke, A., and Noll, T. G. (2007). "Fundamental analysis of resistive nanocrossbars for the use in hybrid nano/cmos-memory," *Proceedings of 33rd ESSCIRC*. 328–331. doi: 10.1109/ESSCIRC.2007.4430310

Froemke, R., and Dan, Y. (2002). Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature* 416, 433–438. doi: 10.1038/416433a

Fukushima, A., Seki, T., Yakushiji, K., Kubota, H., Imamura, H., Yuasa, S., et al. (2014). Spin dice: a scalable truly random number generator based on spintronics. *Appl. Phys. Express*. 7:083001. doi: 10.7567/APEX.7.083001

Garcia, V., Fusil, S., Bouzehouane, K., Enouz-Vedrenne, S., Mathur, N. D., Barthélémy, A., et al. (2009). Giant tunnel electroresistance for non-destructive readout of ferroelectric states. *Nature* 460, 81–84. doi: 10.1038/nature08128

Gerstner, W., Kempter, R., Leo van Hemmen, J., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Lett. Nat.* 383, 76–78. doi: 10.1038/383076a0

Gerstner, W., Ritz, R., and Hemmen, J. L. (1993). Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns. *Biol. Cybern.* 69, 503–515. doi: 10.1007/BF00199450

Goldberg, D. H., Cauwenberghs, G., and Andreou, A. G. (2001). Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons. *Neural Netw*. 14, 781–793. doi: 10.1016/S0893-6080(01)00057-0

Hebb, D. O. (1949). *The Organization of Behavior*. New York: Wiley and Sons.

Henker, S., Mayr, C., Schlüssler, J.-U., Schüffny, R., Ramacher, U., and Heittmann, A. (2007). "Active pixel sensor arrays in 90/65nm CMOS-technologies with vertically stacked photodiodes," in *Proceedings if IEEE International Image Sensor Workshop*. 16–19.

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw*. 17, 211–221. doi: 10.1109/TNN.2005.860850

International Technology Roadmap For Semiconductors (ITRS). (2011). *Emerging Research Devices*.

Jacob, V., Brasier, D. J., Erchova, I., Feldman, D., and Shulz, D. E. (2007). Spike-timing-dependent synaptic depression in the in vivo barrel cortex of the rat. *J. Neurosci*. 27, 1271–1284. doi: 10.1523/JNEUROSCI.4264-06.2007

Jeong, D. S., Kim, I., Ziegler, M., and Kohlstedt, H. (2013). Towards artificial neurons and synapses: a materials point of view. *RSC Adv*. 3, 3169. doi: 10.1039/c2ra22507g

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett*. 10, 1297–1301. doi: 10.1021/nl904092h

Josberger, E. E., Deng, Y., Sun, W., Kautz, R., and Rolandi, M. (2014). Two terminal protonic devices with synaptic like short term depression and device memory. *Adv. Mater*. 26, 4986–4990. doi: 10.1002/adma.201400320

Joubert, A., Belhadj, B., Temam, O., and Heliot, R. (2012). "Hardware spiking neurons design: analog or digital?" in *International Joint Conference Neural Network* (Brisbane, QLD), 1–5. doi: 10.1109/IJCNN.2012.6252600

Kavehei, O. (2013). Highly scalable neuromorphic hardware with 1-bit stochastic nano-synapses. *IEEE Int. Symp. Circuits Syst*. 1648–1651. doi: 10.1109/ISCAS.2014.6865468

Kawahara, A., Azuma, R., Ikeda, Y., Kawai, K., Katoh, Y., Tanabe, K., et al. (2012). "An 8Mb multi-layered cross-point ReRAM macro with 443MB/s write throughput," in *IEEE International Solid-State Circuits Conference* 432–434. doi: 10.1109/ISSCC.2012.6177078

Khan, M., Lester, D., Plana, L., Rast, A., Jin, X., Painkras, E., et al. (2008). "Spinnaker: mapping neural networks onto a massively-parallel chip multiprocessor," in *IEEE International Joint Conference Neural Network* (Hong Kong), 2849–2856. doi: 10.1109/IJCNN.2008.46 34199

Kornijcuk, V., Kavehei, O., Lim, H., Seok, J. Y., Kim, S. K., Kim, I., et al. (2014). Multiprotocol-induced plasticity in artificial synapses. *Nanoscale* 6, 15151–15160. doi: 10.1039/c4nr03405h

Kuzum, D., Jeyasingh, R. G. D., Lee, B., and Wong, H.-S. P. (2012). Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nanoletters* 12, 2179–2186. doi: 10.1021/nl201040y

Kuzum, D., Yu, S., and Wong, H. P. (2013). Synaptic electronics: materials, devices and applications. *Nanotechnology* 24:382001. doi: 10.1088/0957-4484/24/38/382001

Lamprecht, R., and Ledoux, J. (2004). Structural plasticity and memory. *Nat. Rev. Neurosci*. 5, 45–54. doi: 10.1038/nrn1301

Lau, C. N., Stewart, D. R., Williams, R. S., and Bockrath, M. (2004). Direct observation of nanoscale switching centers in metal/molecule/metal structures. *Nano Lett*. 4, 569–572. doi: 10.1021/nl035117a

Lee, H. Y., Chen, P. S.,Wu, T. Y., Chen, Y. S.,Wang, C. C., Tzeng, P. J., et al. (2008). "Low power and high speed bipolar switching with a thin reactive Ti buffer layer in robust HfO2 based RRAM," in *Technology Digital IEEE International Electron Devices Meeting* (San Francisco, CA), 1–4. doi: 10.1109/IEDM.2008.4796677

Lim, H., Kim, I., Kim, J.-S., Hwang, C. S., and Jeong, D. S. (2013). Short term memory of TiO2 based electrochemical capacitors: empirical analysis with adoption of a sliding threshold. *Nanotechnology* 24:384005. doi: 10.1088/0957-4484/24/38/384005

Linares Barranco, B., and Serrano-Gotarredona, T. (2009b). "Exploiting memristance in adaptive asynchronous spiking neuromorphic nanotechnology systems," *9th IEEE Conference Nanotechnology* (Genoa), 601–604.

Linares-Barranco, B., and Serrano-Gotarredona, T. (2009a). Memristance can explain spike-time-dependent-plasticity in neural synapses. *Nat. Proc. NPRE*. doi: 10101/npre.2009.3010.1

Linn, E., Rosezin, R., Kügeler, C., and Waser, R. (2010). Complementary resistive switches for passive nanocrossbar memories. *Nat. Mater*. 9, 403–406. doi: 10.1038/nmat2748

Liu, T.-Y., Yan, T. H., Scheuerlein, R., Chen, Y., Lee, J. K. Y., Balakrishnan, G., et al. (2013). A 130.7 mm^2 2-layer 32Gb ReRAM memory device in 24nm technology. *IEEE Int. Solid-State Circuits Conf*. 432–434, 210–212. doi: 10.1109/ISSCC.2013.6487703

Livi, P., and Indiveri, G. (2009). A current-mode conductance-based silicon neuron for address-event neuromorphic systems. *Int. Symp. Circuits Syst*. 2898–2901. doi: 10.1109/ISCAS.2009.5118408

Lou, X., Gao, Z., Dimitrov, D. V., and Tang, M. X. (2008). Demonstration of multilevel cell spin transfer switching in MgO magnetic tunnel junctions. *Appl. Phys. Lett*. 93, 242502. doi: 10.1063/1.3049617

Marins de Castro, M., Sousa, R. C., Bandiera, S., Ducruet, C., Chavent, A., Auffret, S., et al. (2012). Precessional spin-transfer switching in a magnetic tunnel junction with a synthetic antiferromagnetic perpendicular polarizer. *J. Appl. Phys*. 111, 07C912–07C912-3. doi: 10.1063/1.3676610

Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APS and EPSPS. *Science* 275, 213–215. doi: 10.1126/science.275.5297.213

Masquelier, T., Guyonneau, R., and Thorpe, S. J. (2008). Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE* 3:e1377. doi: 10.1371/journal.pone.0001377

Masquelier, T., Guyonneau, R., and Thorpe, S. J. (2009). Competitive STDP-based spike pattern learning. *Neural Comp*. 21, 1259–1276. doi: 10.1162/neco.2008.06-08-804

Mayr, C., Ehrlich, M., Henker, S., Wendt, K., and Schüffny, R. (2007). Mapping complex, large scale spiking networks on neural VLSI. *Int. J. Appl. Sci. Eng. Technol*. 4, 37–42.

Mayr, C., Noack, M., Partzsch, J., and Schüffny, R. (2010). Replicating experimental spike and rate based neural learning in CMOS. *IEEE Int. Symp. Circuits Systems*. 105–108. doi: 10.1109/ISCAS.2010.5537009

Mayr, C., and Partzsch, J. (2010). Rate and pulse based plasticity governed by local synaptic state variables. *Front. Synaptic Neurosci*. 2:33. doi: 10.3389/fnsyn.2010.00033

Mayr, C., Partzsch, J., Noack, M., Hänzsche, S., Scholze, S., Höppner, S., et al. (2014b). A biological real time neuromorphic system in 28nm CMOS using low leakage switched capacitor circuits. *Trans. Biomed. Circuits Syst*. doi: 10.1109/TBCAS.2014.2379294

Mayr, C., Partzsch, J., Noack, M., and Schüffny, R. (2014a). Configurable analog-digital conversion using the neural engineering framework. *Front. Neurosci*. 8:201. doi: 10.3389/fnins.2014.00201

Mayr, C., Stärke, P., Partzsch, J., Cederstroem, L., Schüffny, R., Shuai, Y., et al. (2012). Waveform driven plasticity in BiFeO3 memristive devices: model and implementation. *Adv. Neural Inf. Process. Syst*. 25, 1700–1708.

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., and Modha, D. S. (2011). "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45 nm," in *Custom Integrative Circuits Conference* (San Jose, CA), 1–4. doi: 10.1109/CICC.2011.6055294

Mu, Y., and Poo, M. M. (2006). Spike timing-dependent LTP/LTD mediates visual experience-dependent plasticity in a developing retinotectal system. *Neuron* 50, 115–125. doi: 10.1016/j.neuron.2006.03.009

Nessler, B., Pfeiffer, M., Buesing, L., and Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput. Biol*. 9:e1003037. doi: 10.1371/journal.pcbi.1003037

Noack, M., Partzsch, J., Mayr, C., and Schüffny, R. (2010). "Biology-derived synaptic dynamics and optimized system architecture for neuromorphic hardware," in *17th International Conference on Mixed Design of Integrated Circuits and Systems* (Warsaw), 219–224.

Ohno, T., Hasegawa, T., Tsuruoka, T., Terabe, K., Gimzewski, J. K., and Aono, M. (2011). Short term plasticity and long term potentiation mimicked in single inorganic synapses. *Nat. Mater*. 10, 591–595. doi: 10.1038/nmat3054

Pershin, Y. V., and Di Ventra, M. (2008). Spin memristive systems: spin memory effects in semiconductor spintronics. *Phys. Rev. B* 78:113309. doi: 10.1103/PhysRevB.78.113309

Querlioz, D., Bichler, O., Dollfus, P., and Gamrat, C. (2013). Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Trans. Nanotechnol*. 12, 288–295. doi: 10.1109/TNANO.2013.2250995

Querlioz, D., Dollfus, P., Bichler, O., and Gamrat, C. (2011). "Learning with memristive devices: how should we model their behavior?" in *7th IEEE/ACM International Symposium on Nanoscale Architectures* (San Diego, CA), 150–156.

Rangan, V., Ghosh, A., Aparin, V., and Cauwenberghs, G. (2010). A subthreshold aVLSI implementation of the Izhikevich simple neuron model. *Conf. Proc. IEEE Eng. Med. Biol. Soc.* 2010, 4164–4167. doi: 10.1109/IEMBS.2010.5627392

Shuai, Y., Ou, X., Luo, W., Du, N., Wu, C., Zhang, W., et al. (2013). Nonvolatile multilevel resistive switching in Ar+ irradiated BiFeO3 thin films. *IEEE Electron Device Lett.* 34, 54–56. doi: 10.1109/LED.2012.2227666

Shuai, Y., Zhou, S., Wu, C., Zhang, W., Bürger, D., Slesazeck, S., et al. (2011). Control of rectifying and resistive switching behavior in bifeo3 thin films. *Appl. Phys. Express* 4, 095802. doi: 10.1143/APEX.4.095802

Sjöström, J., and Gerstner, W. (2010). Spike-timing dependent plasticity. *Scholarpedia* 5:1362. doi: 10.4249/scholarpedia.1362

Snider, G. S. (2008). "Spike-timing-dependent learning in memristive nanodevices," in *IEEE International Symposium Nano Architecture* (Anaheim, CA), 85–92. doi: 10.1109/NANOARCH.2008.4585796

Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *Nature* 453, 80–83. doi: 10.1038/nature06932

Suri, M., Querlioz, D., Bichler, O., Palma, G., Vianello, E., Vuillaume, D., et al. (2013). Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Trans. Electron Devices* 60, 2402–2409. doi: 10.1109/TED.2013.2263000

Upadhyaya, H. M., and Chandra, S. (1995). Polarity dependent memory switching behavior in Ti/Cd Pb S/Ag system. *Semicond. Sci. Technol.* 10, 332–338. doi: 10.1088/0268-1242/10/3/016

Varela, J., Sen, K., Gibson, J., Fost, J., Abbott, L. F., and Nelson, S. B. (1997). A quantitative description of short term plasticity at excitatory synapses in layer 2/3 of rat visual cortex. *J. Neurosci.* 17, 7926–7940.

Vincent, A. F., Locatelli, N., Zhao, W. S., Klein, J.-O., Galdin-Retailleau, S., and Querlioz, D. (2015). Analytical macrospin modeling of the stochastic switching time of spin transfer-torque magnetic tunnel junctions. *IEEE Trans. Electron Devices* 62, 164–170. doi: 10.1109/TED.2014.2372475

Vincent, A., Larroque, J., Zhao, W. S., Ben Romdhane, N., Bichler, O., Gamrat, C., et al. (2014). "Spin-transfer torque magnetic memory as a stochastic memristive synapse," in *IEEE International Symposium Circuits System* (Melbourne, VIC), 1074–1077. doi: 10.1109/ISCAS.2014.6865325

Wang, X., Chen, Y., Xi, H., Li, H., and Dimitrov, D. (2009). Spintronic memristor through spin-torque-induced magnetization motion. *IEEE Electron Device Lett.* 30, 294–297. doi: 10.1109/LED.2008.2012270

Waser, R., and Aono, M. (2007). Nanoionics-based resistive switching memories. *Nat. Mater.* 6, 833–840. doi: 10.1038/nmat2023

Wijekoon, J., and Dudek, P. (2008). Compact silicon neuron circuit with spiking and bursting behavior. *Neural Netw.* 21, 524–534. doi: 10.1016/j.neunet.2007.12.037

Wong, H.-S. P., Lee, H.-Y., Yu, S., Chen, Y.-S., Wu, Y., Chen, P.-S., et al. (2012). Metal oxide RRAM. *Proc. IEEE* 100, 1951–1970. doi: 10.1109/JPROC.2012.2190369

Wu, X., Zhou, P., Li, J., Chen, L. Y., Lu, H. B., Lin, Y. Y., et al. (2007). Reproducible unipolar resistance switching in stoichiometric ZrO2 films. *Appl. Phys. Lett.* 90, 183507. doi: 10.1063/1.2734900

Yamada, H., Garcia, V., Fusil, S., Boyn, S., Marinova, M., Gloter, A., et al. (2013). Giant electroresistance of super-tetragonal BiFeO3-based ferroelectric tunnel junctions. *ACS Nano*. 7, 5385–5390. doi: 10.1021/nn401378t

Yang, R., Terabe, K., Yao, Y., Tsuruoka, T., Hasegawa, T., Gimzewski, J. K., et al. (2013). Synaptic plasticity and memory functions achieved in a WO3-x based nanoionics device by using the principle of atomic switch operation. *Nanotechnology* 24, 384003. doi: 10.1088/0957-4484/24/38/384003

You, T., Du, N., Slesazeck, S., Mikolajick, T., Li, G., Bürger, D., et al. (2014). Bipolar electric-field enhanced trapping and detrapping of mobile donors in $BiFeO_3$ memristors. *ACS Appl Mater Interfaces* 6, 19758–19765. doi: 10.1021/am504871g

Young, J. M. (2007). Cortical reorganization consistent with spike timing-but not correlation-dependent plasticity. *Nat. Neurosci.* 10, 887–895. doi: 10.1038/nn1913

Yu, S., Gao, B., Fang, Z., Yu, H., Kang, J., and Wong, H.-S. P. (2013). Stochastic learning in oxide binary synaptic device for neuromorphic computing. *Front. Neurosci.* 7:186. doi: 10.3389/fnins.2013.00186

Zamarreño-Ramos, C., Camuñas-Mesa, L. A., Pérez-Carrasco, J. A., Masquelier, T., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). On spike-timing-dependentplasticity, memristive devices, and building a self-learning visual cortex. *Front. Neurosci.* 5:26. doi: 10.3389/fnins.2011.00026

Zhang, L., Tao, H., Holt, C., Harris, W., and Poo, M. (1998). A critical window for cooperation and competition among developing retinotectal synapses. *Nature* 395, 37–44. doi: 10.1038/25665

Zhang, Y., Zhao, W., Klein, J.-O., Kang, W., Querlioz, D., Zhang, Y., et al. (2014). "Spintronics for low-power computing," in *Design, Automation and Test in Europe Conference and Exhibition* (Dresden), 1–6. doi: 10.7873/DATE.2014.316

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 21 October 2014; accepted: 05 February 2015; published online: 02 March 2015.

Citation: Saïghi S, Mayr CG, Serrano-Gotarredona T, Schmidt H, Lecerf G, Tomas J, Grollier J, Boyn S, Vincent AF, Querlioz D, La Barbera S, Alibart F, Vuillaume D, Bichler O, Gamrat C and Linares-Barranco B (2015) Plasticity in memristive devices for spiking neural networks. Front. Neurosci. ***9****:51. doi: 10.3389/fnins.2015.00051*

This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.

Tunnel junction based memristors as artificial synapses

Andy Thomas[1, 2*], *Stefan Niehörster*[1], *Savio Fabretti*[1], *Norman Shepheard*[1, 3], *Olga Kuschel*[4], *Karsten Küpper*[4], *Joachim Wollschläger*[4], *Patryk Krzysteczko*[1, 5] and *Elisabetta Chicca*[3]

[1] Thin Films and Physics of Nanostructures, Bielefeld University, Bielefeld, Germany, [2] IFW Dresden, Institute for Metallic Materials, Dresden, Germany, [3] Cognitive Interaction Technology Center of Excellence and Faculty of Technology, Bielefeld University, Bielefeld, Germany, [4] Fachbereich Physik and Center of Physics and Chemistry of New Materials, Osnabrück University, Osnabrück, Germany, [5] Physikalisch Technische Bundesanstalt, Braunschweig, Germany

We prepared magnesia, tantalum oxide, and barium titanate based tunnel junction structures and investigated their memristive properties. The low amplitudes of the resistance change in these types of junctions are the major obstacle for their use. Here, we increased the amplitude of the resistance change from 10% up to 100%. Utilizing the memristive properties, we looked into the use of the junction structures as artificial synapses. We observed analogs of long-term potentiation, long-term depression and spike-time dependent plasticity in these simple two terminal devices. Finally, we suggest a possible pathway of these devices toward their integration in neuromorphic systems for storing analog synaptic weights and supporting the implementation of biologically plausible learning mechanisms.

Keywords: memristors, artificial synapses, tunnel junction, synaptic plasticity, neuromorphic systems

Edited by:
Themis Prodromakis, University of Southampton, UK

Reviewed by:
Joaquin Sitte, Queensland University of Technology, Australia
Sukru Burc Eryilmaz, Stanford University, USA

Correspondence:
Andy Thomas, Thin Films and Physics of Nanostructures, Department of Physics, Universitaetsstr. 25, Bielefeld 33615, Germany
andy.thomas@uni-bielefeld.de

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *20 October 2014*
Accepted: *24 June 2015*
Published: *07 July 2015*

Citation:
Thomas A, Niehörster S, Fabretti S, Shepheard N, Kuschel O, Küpper K, Wollschläger J, Krzysteczko P and Chicca E (2015) Tunnel junction based memristors as artificial synapses. Front. Neurosci. 9:241. doi: 10.3389/fnins.2015.00241

1. Introduction

Memristors have attracted great interest for a variety of applications in recent years (Prezioso et al., 2015). An obvious use would be as a memory device (Chen et al., 2008; Linn et al., 2010; Lee et al., 2011) or, more ambitiously, a reconfigurable logic device (Borghetti et al., 2009; Xia et al., 2009; Borghetti et al., 2010; Muenchenberger et al., 2011; Yan et al., 2011). However, the arguably most interesting implementation of memristive devices is neuromorphic computing (Jo et al., 2010; Indiveri et al., 2013).

Neuromorphic engineering is a relatively young research field, which was originally proposed by Mead (1989, 1990) in the late 80s. Neuromorphic devices and architectures are designed to emulate the style of computation of biological systems and exploit biological strategies for optimizing robustness to noise and fault tolerance, as well as maximizing compactness and minimizing power consumption. Nevertheless, the most attractive feature of biological systems is their ability to learn and adapt to new situations. Artificial agents equipped with such abilities would have a broad range of applications. The possibility of embedding learning capabilities in neuromorphic systems is therefore extremely appealing. One route toward implementing these synaptic weights is the memristor. However, well-characterized materials suitable for the construction of memristive devices with large memristive switching are needed. In this manuscript we investigate possible solutions to this problem.

A possible realization of a memristive device is a metal-insulator-metal structure. In particular, this can be a tunnel junction. Then, a 1–3 nm thin insulator separates two metal electrodes, and the tunneling current is determined while the bias voltage is applied. The scalability of (magnetic) tunnel junctions was already shown in magnetic random access memory devices, and a 16 Mb chip is commercially available. Consequently, we suggest the use of memristive tunnel junctions as artificial synapses in neuromorphic circuits.

In this manuscript, we will present several oxides used as the barrier materials in tunnel junctions. A cartoon of the layer stacks is depicted in **Figure 1**. Many oxides can exhibit memristive switching behavior in such a tunneling structure, which allows us to tailor the electrode and barrier materials for a given application. First, we introduce magnesia based tunnel junctions and use them to look into the analogs of long-term potentiation, long-term depression, and spike-time dependent plasticity, which are the basic functional properties of biological synapses (Thomas, 2013). However, the resistance change in these junctions is rather small [$(R_{\max} - R_{\min})/R_{\min} = 8\%$]. Consequently, new material combinations are tested, and we will show resistive switching in $BaTiO_3$ and Ta-O based systems. $BaTiO_3$ and TaO based junctions exhibit up to 10 times larger resistance changes, which improves the prospects of these systems in semiconductor-based neuromorphic circuits. The general suitability of these junctions is discussed in the last section and compared to the requirements suggested by Indiveri et al. (2013).

2. Materials and Methods

2.1. Magnesia Based Tunnel Junctions

The MgO-based magnetic tunnel junctions are sputtered on SiO_2 generated by thermal oxidation of Si. The bottom layer stack consists of Ta/Cu—N/Ta/Pt—Mn/Co—Fe/Ru, the functional tunnel system is composed of Co—Fe—B/MgO/Co—Fe—B, while Ta/Cu/Ru form the top layers. The deposition is performed at room temperature and followed by a post-annealing step at 360 °C for 90 min. Afterwards, elliptical tunnel junctions with a major (minor) axis of 350 nm (150 nm) are prepared by successive steps of electron beam lithography and ion beam etching. More details on the preparation and characterization of magnesia based tunnel junctions are given in previous publications (Krzysteczko et al., 2008, 2009).

All measurements were carried out at room temperature with a voltage source. The voltage pulses of up to 800 mV have 1 s duration and lead to a current density of 1×10^6 A/cm^2 to 10×10^6 A/cm^2. These values are close to the dielectric breakdown voltage of the devices. Transmission electron microscopy images of MgO junctions before and after the dielectric breakdown are presented in our previous work (Thomas et al., 2008; Schaefers et al., 2009). Consequently, the investigations utilizing sequences of voltage pulses were limited to a maximum voltage of 500 mV. The resistance of the device is determined 200 ms after the write pulse by measuring the current at a voltage of 20 mV. The base resistance of the layer stack is approximately 35 $\Omega\mu m^2$ leading to a resistance of approximately 200 Ω for the given lateral junction size. The current flows from the top to the bottom electrode at positive bias voltages.

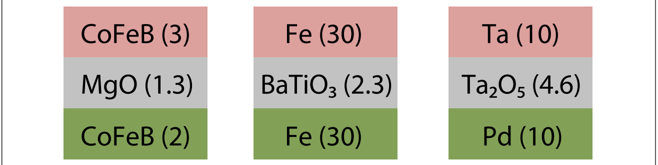


FIGURE 1 | Typical functional layer stacks of the presented memristive tunnel junctions. The thickness of the individual layers are given in parentheses in nm.

2.2. Barium Titanate Junctions

The preparation of barium titanate (BTO) is more challenging than the other materials, because its tetragonal phase with perpendicular orientation of the *c*-axis is required. Therefore, suitable substrate and electrode materials are essential. The lattice constant of MgO is 4.21 Å (Rocksalt structure), while that of iron is 2.86 Å (bcc). Because of the cubic lattice structure of iron and magnesia, the lattice mismatch is 4% with respect to two unit cells of Fe in the diagonal direction to one unit cell of MgO [Fe(001);<110>‖MgO(001);<100>]. Additionally, the lattice mismatch between Fe and BTO (001) is 1.4% in the diagonal growing case, making this system a good candidate for coherent tunneling junctions. Therefore, we chose Fe/BTO/Fe tunneling junctions prepared by rf-magnetron sputtering.

Because of both the small lattice mismatch between Fe and MgO and reliable tunneling junctions of Fe/BTO/Fe, we chose MgO (100) substrates. BTO films were prepared by rf-magnetron sputtering from a single BTO-target (3 inch) with an applied power of 50 W and an argon pressure of 2.1×10^{-3} mbar. The substrate temperatures T_S during the BTO deposition was varied from room temperature up to 918 °C. The crystal structure was investigated by X-ray diffraction (XRD) using Cu-K$_\alpha$ radiation, and the film thickness was measured by X-ray reflectrometry (XRR). After deposition the composition was measured by XPS (Ba 21%, Ti 21%, O 58%) for $T_S = 737$ °C and is close to the stoichiometric ratio of 1:1:3. The tunneling junctions were structured with standard optical lithography techniques. The junction area of the samples presented in this manuscript is 25 µm×25 µm. The junction resistance was approximately 4 kΩ at an applied voltage of 10 mV.

Initially, we characterized our BTO films directly sputtered on MgO by XRD. The film thickness was determined by XRR to calibrate the sputtering. Then, the thickness was set to 10 nm for all samples.

Figure 2 shows the XRD patterns of the BTO films deposited at the given temperature T_S. Both the BTO (002) and the BTO (004) peak intensities increase with increasing temperature up to 918 °C. Furthermore, the position of the BTO (002) peak shifts from 44.425° at $T_S = 689$ °C to 45.925 °C at $T_S = 918$ °C, corresponding to a decrease in the *c*-axis lattice constant from 4.078 Å down to 3.954 Å, which is shown in detail in **Table 1**.

The *c*-axis lattice parameter of the sample at $T_S = 689$ °C exhibits a lattice constant of $c = 4.035$ Å, which closely agrees with the c-axis lattice parameter of $c = 4.036$ Å (Kim et al., 1995). Four phases are possible in BTO: rhombohedric, orthorhombic, tetragonal, and cubic. We can assume that we have achieved a tetragonal phase with perpendicular orientation of the *c*-axis,

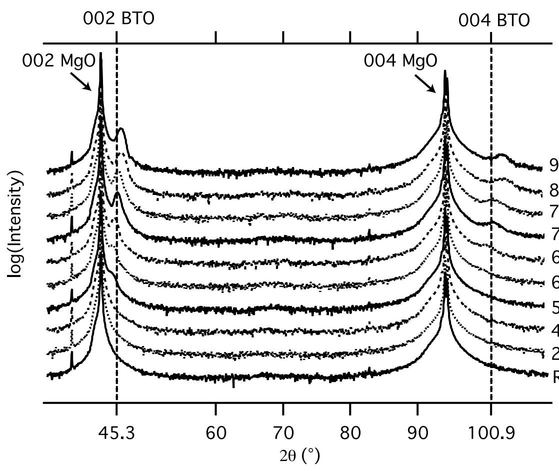


FIGURE 2 | X-ray diffraction pattern of the barium titanate films at different deposition temperatures on MgO substrates. An increase of the 002 and 004 BTO-peaks is observed with increasing substrate temperature.

TABLE 1 | The *c*-axis lattice parameters of BTO (10 nm) layers on MgO substrates calculated via the 002 peak at different deposition temperatures.

T [°C]	595	642	689	737	786	830	918
c [Å]	4.078	4.062	4.035	3.997	3.989	3.951	3.954

At RT, 261 and 452° no peak is visible in the spectra.

which is in good agreement with results previously reported by Kim et al., who fabricated epitaxial BTO films onto MgO (001) substrates (Kim et al., 1995).

2.3. Tantalum Oxide

All films are fabricated by dc magnetron sputter deposition with a base pressure of 3.5×10^{-7} mbar and a sputter pressure of 1.3×10^{-3} mbar. The junctions are defined by optical lithography and argon ion beam etching leading to a size of $10 \times 10\,\mu$m.

We used Pd as the bottom and Ta as the top electrode to generate asymmetric barrier interfaces, both electrodes with a thickness of 10 nm. Our 4.6 nm Ta-O film was produced by plasma oxidation of a 2 nm Ta film as previously reported by Park and Im (1992). The penetration depth is regulated by the bias voltage of the plasma (Rottländer et al., 2001; Thomas et al., 2003). The oxygen concentration was regulated by the oxidation time to generate a tunneling barrier with a high concentration of oxygen vacancies. The generation and movement of oxygen vacancies at one interface in an electrical field results in the resistance change (Krzysteczko et al., 2009; Yang et al., 2010).

3. Results and Discussion

3.1. Magnesia Based Tunnel Junctions

The resistance change of the MgO junctions is determined by a number of voltage pulses, leading to a relative change in resistance as depicted in **Figure 3A**. First, we will look into one example, where a pulse sequence of 30 voltage pulses is used, which corresponds to the bars of **Figure 3A** marked with digits i and 1–4. The pulse sequence itself is described by the cartoon in **Figure 3B** and is inspired by biological data (Rose and Dunwiddie, 1986). The initial state is indicated by the green bar marked with an i.

Now, one pulse sequence (30 voltage pulses) of the described shape is applied to the tunnel junction, and the resistance increases to the value indicated by the digit 1. An additional pulse sequence further increases the resistance to the value signified by the digit 2. Subsequently, the resistance was increased by several pulse sequences finally leading to the value indicated by the digit 4. We observe a systematic decrease of the resistance change after every pulse, and eventually the resistance saturates and reaches its maximum or minimum value. This is true for all memristive devices (Chua, 2014).

This was also performed for pulse sequences with a different number of voltage pulses. The number of voltage pulses in each pulse sequence is given by the abscissa in **Figure 3A**. We carried out these measurements for negative voltages as depicted in **Figure 3B**, leading to a resistance increase, which corresponds to the blue bars in **Figure 3A**. A resistance decrease is caused by positive voltages, which corresponds to the orange bars in **Figure 3A**. The maximum voltage was kept constant at $|v_{\max}| = 500$ mV in all cases.

The increase (decrease) in the conductivity of the junctions can be associated with long-term potentiation LTP (depression, LTD) in a biological neural network. Linares-Barranco et al. suggested to shape the pulses in a particular way (see Linares-Barranco and Serrano-Gotarredona, 2009): The increasing and decreasing edges of the pulses follow an exponential increase and decay, respectively. Two subsequent pulses can generate a positive as well as negative net flux. If we now take similar data depending on the spike-timing, we acquire the flux-dependent plasticity shown in **Figure 4**.

If we fit the collected data, we obtain

$$G(\varphi) = \frac{1}{148\,\text{mS} + 3.77\,\text{mS} \cdot \exp(-\varphi/46.3\,\text{Vs})}$$

for positive as well as

$$G(\varphi) = 6.6 \times 10^{-3}\,\text{mS} + 0.177 \times 10^{-3}\,\text{mS} \cdot \exp(\varphi/47\,\text{Vs})$$

for negative values, where G denotes the conductance and φ denotes the flux. This leads to the red fitting curves in **Figure 4** and can be used to calculate the relative change in conductance also depicted in **Figure 4** in blue. The same kind of functions can be used to fit biological data, e.g., by Bi et al. (Bi and Poo, 1998). Therefore, we are able to show behavior similar to spike-timing dependent plasticity (STDP) in these simple two terminal devices (Jo et al., 2010; Krzysteczko et al., 2012). In the following sections, we suggest more barrier materials to be used in memristive tunnel junctions. We aim to find larger resistance changes than up to 8% in magnesia based tunnel junctions. Furthermore, we would like to show that memristive switching is frequently observed in these kinds of structures.

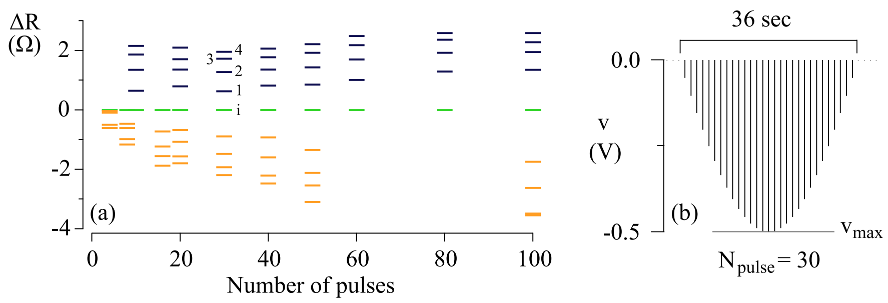


FIGURE 3 | (A) Long-term potentiation and long-term depression of an MgO-based magnetic tunnel junction. The refreshed state is set to zero (green bars). The relative resistance increase is shown in blue and the relative resistance decrease in orange. **(B)** A cartoon of an example pulse sequence of 30 pulses. It consists of a series of 1 s rectangular pulses, convoluted by a sinusoidal half-wave with the amplitude v_{max}. The pulses are separated by 200 ms intervals. For similar pulse sequences, the pulse widths and pulse intervals are always fixed (1 s and 0.2 s, respectively) and the sine function involved in the convolution has a half-period given by the total duration of pulse sequence.

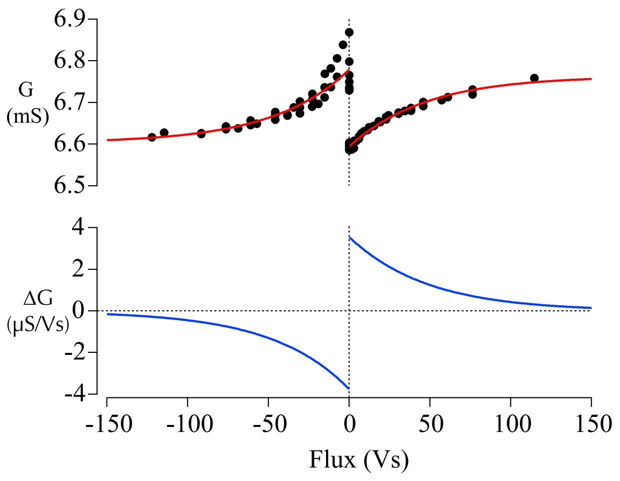


FIGURE 4 | Flux-dependent plasticity of memristive magnetic tunnel junctions. Top: The asymmetric conductivity of a memristive tunnel junction. Positive flux is associated with causal spike-timing, negative flux with anti-causal spike-timing. The asymmetry typical for stdp is evaluated by the fitting curves. **Bottom**: The derivative of the fitting curves in the top graph is calculated to provide a measure of the change in synaptic strength.

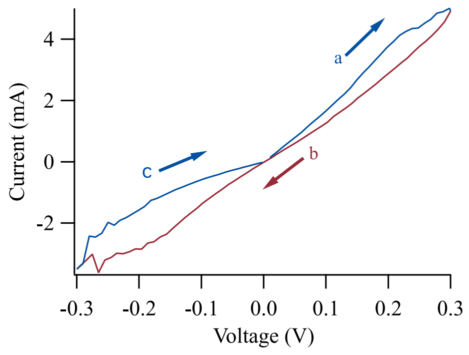


FIGURE 5 | Current-voltage characteristics of a BTO based tunnel junction. The measurement sequence is a, b, c at an applied voltage range of −300 to 300 mV. The hysteresis loop shows a type II non-crossing hysteresis.

3.2. Barium Titanate Junctions

Initial success with memristive BTO tunnel barriers was published by Chanthbouala et al. on LSMO/BTO/Co systems, in which junction preparation was accomplished by pulsed laser deposition (Chanthbouala et al., 2012). Therefore, we looked into sputtered memristive tunnel junctions based on BTO barriers (Von Hippel, 1950; Kim et al., 1995). In other contexts, this compound is best known for its ferroelectric properties, which are caused by the tetragonal crystal structure. In the future, this might lead to multi-functional tunnel junctions (Fiebig, 2005; Tsymbal, 2006), i.e., junctions exhibiting memristive behavior and ferroelectricity at the same time.

Figure 5 shows a tunneling hysteresis loop in the applied dc bias range of −300 to 300 mV for $T_S = 737\,^{\circ}\mathrm{C}$. The measured sequence was from 0 mV up to 300 mV, down to −300 mV and back to 0 mV. The time between each data point was 200 ms. The figure displays the pinched hysteresis loop characteristic for all memristors (Chua, 2014). According to the theoretical overview of Pershin et al., memristors can be categorized into two types (Pershin and Di Ventra, 2011): Self crossing and non-self-crossing, i.e., the two branches of the hysteresis loop do or do not cross each other.

For BTO, we are interested in the unusual non-crossing hysteresis loop, since most memristive systems show a crossing, i.e., a type-I behavior. A non-self-crossing hysteresis loop can be observed in thermistors and elastic memcapacitive systems. In contrast, Chanthbouala et al. observed a crossing (type-I) I-V-curve for a Lsuppa-Sr-Mn-O/BTO/Co system (Chanthbouala et al., 2012).

However, the memristive effect is very sensitive to the preparation of the junctions, as observed and discussed by Krzysteczko et al. for MgO-based systems (Krzysteczko et al., 2008, 2012). Our BTO system consists of Fe electrodes and is prepared by magnetron sputtering. Chanthbouala et al. grew their junctions using pulsed laser deposition and discussed the influence of structure on the nucleation and propagation of domain walls. The Kolmogorov-Avrami-Ishibashi model describes clean (epitaxial) systems, in which switching is predominantly caused by domain wall propagation (Ishibashi and Takagi, 1971; Hashimoto et al., 1994; Jo et al., 2009). In disordered systems, nucleation-limited switching models should be used (Du and Chen, 1998; Tagantsev et al., 2002), which might explain the different crossing behaviors, even for similar systems utilizing BTO tunnel barriers. The combination of crossing as well as non-crossing behavior might be exploited for compact sequential logics as suggested by You et al. (2014).

BTO-based memristive systems are promising because of their large amplitude of resistive switching. **Figure 5** exhibits an 80% change in the tunneling current at −200 mV, which is approximately 10 times larger than the amplitude in our previously investigated MgO-based tunnel junctions (Krzysteczko et al., 2008, 2012). Furthermore, the epitaxial BTO systems indicate the potential of the BTO tunneling systems that exhibit resistance changes of a factor of 750 for 3 nm films (Garcia et al., 2009). However, the epitaxial growth might pose additional challenges for the integration in existing neuromorphic circuits.

3.3. Tantalum Oxide

Previous reports of Ta-O-based memristive devices show a fast and stable switching behavior for at least 1×10^{10} cycles (Yang et al., 2010; Torrezan et al., 2011). However, the thickness of the Ta-O layer always exceeded 7 nm. Consequently, we prepared tunnel junction type systems with a Ta-O barrier.

The samples with oxidation times of 150 s and 200 s reached the highest ratio between the lowest and highest resistance states. **Figure 6** shows an I-V-loop of a junction oxidized for 150 s, we observed values of up to 80% in the depicted junction. We swept the voltage from zero to −600 to 600 mV and back to zero. Voltages of more than 600 mV led to a dielectric breakdown of the junctions (Thomas et al., 2008; Schaefers et al., 2009). All measurements are done with the bottom electrode as the reference potential.

A Brinkman-Fit (Brinkman et al., 1970) of our measurements shows a Ta-O barrier with an effective thickness of 1.7–1.9 nm, a height in the range of 0.90–1.12 eV and an asymmetry in the range of 0.08–0.35 eV. The difference between the effective barrier thickness and the measured thickness of the Ta-O film seems to result from the post-sputtering *in-situ* oxidation, which generates a rough interface to the bottom electrode.

Furthermore, we are able to reach more than two states in a Ta-O based tunnel junctions, as shown in **Figure 7**. We generate the resistance steps by applying a voltage of ±600 mV for 15 s. The resulting resistance levels are measured with a voltage of 10 mV for 180 s. The first three positive (green) voltage pulses increase the resistance while the last (blue) negative pulse decreases it. We can observe the analog of long-term depression and long-term potentiation in Ta-O based junctions, and we have increased the signal by more than a factor of 10 compared to 10% resistance change in MgO based systems. This could emulate the synaptic weight in a neuromorphic chip, and a possible future implementation is suggested in the fifth section.

4. Alternate Mechanisms for Memristive/Resistive Switching

There are several other mechanisms that can lead to memristive or resistive switching in mesoscopic systems (Waser et al., 2009): Nanomechanical effects, molecular switching, electrostatic/electronic effects, electrochemical metallization, valence change, thermochemical effects, phase change, magnetoresistance, ferroelectricity, and the presented change in the effective tunnel barrier thickness. In the next paragraphs, we will present a few published results based on different mechanisms. However, this is not an extensive review of the current state of the art, but rather a comparison of several differences and similarities to emphasize the current challenge for memristive tunnel junctions: The amplitude of the resistance change.

Redox-related chemical effects—which include electrochemical metallization, valence change, and thermochemical effects—were intensively studied (Waser and Aono, 2007), even before the term *memristor* moved back into the focus of attention (Strukov et al., 2008). A lot of the research interest was driven by the search for a non-volatile memory that could replace Flash at some point in the future (Waser, 2008).

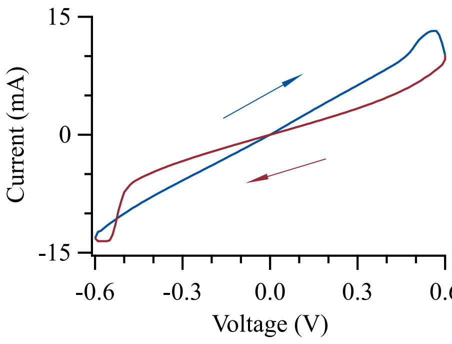


FIGURE 6 | Memristive switching of Pd-TaO-Ta tunnel junction. The barrier was oxidized for 150 s leading to the largest ratio between the high and low resistance states.

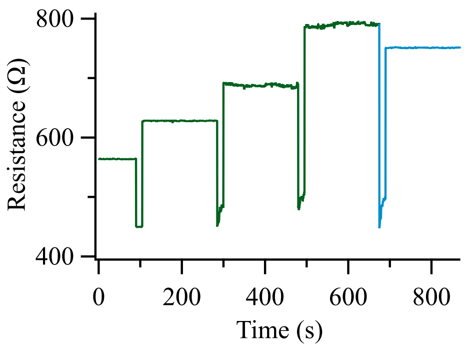


FIGURE 7 | Analogs of long-term potentiation and long-term depression of a Ta-O based tunnel junction. Positive voltage pulses are depicted in green, negative voltage pulses in blue. The pulse voltage was ±600 mV in all cases.

We start with systems based on electrochemical metallization. These devices consist of a trilayer of an electrochemically inert material, an electrochemically active material and a thin film electrolyte sandwiched in between the two (Kozicki and Mitkova, 2008). Sometimes, this mechanism is also called conductive bridging or programmable metallization cell. An applied voltage leads to a formation or dissolution of a metal filament between the two electrodes, decreasing or increasing the resistance of the device, respectively. The direction of the process depends on the polarity of the applied voltage, which is consequently denoted as bi-polar switching. An $R_{\text{off}}/R_{\text{on}}$ ratio [resistance in the on (R_{on}) and off (R_{off}) states] of more than 1×10^5 and switching times of less than 100 ns were reported (Waser et al., 2009). However, often a forming step of voltages on the order of 5 V is required before the switching voltage of approximately 1 V can be used.

The change of the resistance in a device can also be induced by thermal effects. These mechanisms do not depend on the direction of the current flow. Therefore, these mechanisms always have unipolar characteristics. Two prominent examples are thermochemical switching and phase change materials (Kuzum et al., 2012). The first case can be observed in, e.g., transition metal oxides. Gibbons and Beadle investigated this already in the 1960s in Ni-O (Gibbons and Beadle, 1964). The on state is also caused by a filament connecting the two electrodes (Waser et al., 2009) and an on/off ratio of two orders of magnitude is reported, e.g., for Pt/Ni-O/Pt devices (Yun et al., 2007). In the second case, a structural phase change causes the resistance change. Simpson et al. report, e.g., on $GeTe/Sb_2Te_3$ devices (Simpson et al., 2011). The resistance change of two orders of magnitudes was induced by voltage pulses of several volts for 50–100 ns. Similar resistance changes were reported by Eryilmaz et al., although the amplitude of this change decreases to 100% if a continuously varying resistance is desired (Eryilmaz et al., 2014).

The research investigating magnetic as well as ferroelectric systems was looking into their use as magnetic or ferroelectric random access memory, i.e., into bi-stable systems comparable to the research of redox-related chemical effects. Magnetic random access memory is often based on magnetic tunnel junctions (MTJs). In MTJs, two ferromagnetic electrodes are separated by a thin insulating layer. Then, tunnel magnetoresistance can be observed, i.e., the resistance in these devices is small/large if the magnetization of two ferromagnetic layers are aligned in parallel/ antiparallel, respectively (Julliere, 1975; Moodera and Mathon, 1999). The alignment of the softer magnetic layer can be switched by current pulses because the spin of the tunneling electrons is flipped and leads to a torque, eventually switching the magnetization (Huai et al., 2004). Magnesia based MTJs exhibit on/off ratios of 2, switching voltages of approximately 500 mV and switching times in the order of ns (Kishi et al., 2008; Schaefers et al., 2009).

In the following, we try to list some of the requirements for our memristive devices. Bipolar switching is preferred, because it allows to attain STDP functionality by simple pulse shaping and overlapping (Linares-Barranco and Serrano-Gotarredona, 2009). We used the suggested pulse shaping to exemplarily demonstrate STDP using the MgO-based MTJs. The same scheme is valid for the other systems based on Ta-O. However, we increased the on-off ratio to 100% to improve the performance of the devices.

The access to a continuously varying resistance or at least multiple states would also permit the representation of synaptic weight by a single device. In either case, a large on/off ratio is desired. A write voltage of less than 3.3 V (for e.g., 350 nm technology) would be advantageous to be compatible to existing neuromorphic circuits, ideally without the requirement of a forming step. The condition that small voltages should not lead to a resistance change would be a favorable deviation from the ideal memristive behavior. Otherwise, every read process would change the resistance value of the memristive system. Finally, the memristive devices should be scalable down to nanoscopic dimensions.

If we compare the memristive tunnel junctions to these requirements, we observe bipolar switching and the access to a continuously varying resistance. The write voltages are in the order of 500 mV and no forming step is required. The junctions exhibit a voltage threshold for the resistance change and the tunnel junctions can be prepared to very small lateral dimensions. 50×50 nm were demonstrated already 10 years ago, and similar structures are the basis of commercially available magnetic RAM (Kubota et al., 2003). This is comparable to other technologies such as electrochemical metallization where the scalability has been demonstrated down to devices with diameters of 20 nm (Valov et al., 2011).

The amplitude of the resistance change is one obvious disadvantage of the memristive tunnel junctions if compared to the other mechanisms with on/off ratios of up to 5 orders of magnitude. Our experiments with magnesia based junctions exhibited resistance changes of less than 10%. Here, we tested new material combinations and increased the resistance change from 10% in MgO to 100% in $BaTiO_3$ and Ta-O to tackle the most obvious challenge for memristive tunnel junctions as synaptic weights in future neuromorphic circuits.

5. Possible Integration in Neuromorphic Systems

An ambitious aim is the full implementation of a memristive layer stack on top of a functional neuromorphic circuit. In a step toward this goal, we first contrast the different tunnel barrier materials presented in this manuscript to each other.

Magnesia was the first material where we observed memristive behavior in MTJs (Krzysteczko et al., 2008). The memristive tunnel junctions exhibited key features mimicking synaptic plasticity such as long-term depression, long-term potentiation and STDP (Krzysteczko et al., 2012; Thomas, 2013). However, the maximum amplitude between the lowest and highest resistance is 8%. This limits the use in actual devices, as discussed in the next paragraph. The main goal of the research presented in this manuscript is the preparation of memristive tunnel junctions showing larger resistance changes while maintaining the key features. Consequently, tantalum oxide and barium titanate are discussed in the following paragraphs.

Both $BaTiO_3$ and Ta-O exhibit a resistance change of approximately 80% and allow access to a continuously variable property (resistance in our case), which can be used as the synaptic strength in a future device. Initially, $BaTiO_3$ was chosen as one barrier material, because of the results published by Chanthbouala et al. (2012). However, the preparation process was discussed in detail, indicating a high-temperature (700 °C) process. The high temperatures might complicate a full integration of the $BaTiO_3$ memristors on top of existing CMOS technology, which was discussed for resistive RAM/resistive switching earlier (e.g., Pinnow and Mikolajick, 2004; Pan et al., 2014). Therefore, we focus on the Ta-O based devices.

A possible integration of memristor based devices with neuromorphic synaptic circuits was suggested by Indiveri et al. (2013). In the following, we compare the requirements pointed out in Indiveri et al. (2013) with the properties of the Ta-O based junctions. The suggested voltage is comparable to the voltage applied to our junctions. The impedance change of the memristors was assumed to be between 1 kΩ and 7 kΩ (i.e., a factor of 7) and exhibit 4 discrete resistance states, although 2 states would also be possible (Brader et al., 2007; Mitra et al., 2009). Our Ta-O devices show a resistance change of a factor of 2, which is approaching the necessary amplitudes at least for 2 resistance states. The absolute resistance of a tunnel junction is determined by the barrier thickness (with an exponential relationship) as well as the junction's area (with a linear relationship). This allows for simple tuning of the junction's resistance to the desired value by changing the junction area and barrier thickness according to the requirements of the circuit design and area constraints.

To further demonstrate the possible integration of Ta-O memristors into existing technologies, we designed a neuromorphic chip comprised of synaptic and neural circuits as well as various test structures for the deposition of memristor devices (pads marked with red frames in **Figure 8**). The chip was fabricated using a standard AMS 0.35 μ m CMOS process, covers an area of about 1.6 mm^2 and includes neuromorphic circuits as described in Chicca et al. (2014). The test structures enable the deposition of the presented layer stacks on top of the chip and subsequent e-beam lithography as well as ion beam etching to define the junctions. The underlying synaptic circuits are the ones proposed by Indiveri et al. (2013) and this scheme supports the direct integration of the memristor by allowing the implementation of programmable synaptic weights. The corresponding simulation results are also given by the same authors (Indiveri et al., 2013).

Consequently, we will fully characterize the response of the synaptic and neuronal circuit for several memristor states with the goal of validating in hardware the integration, which is so far only supported by simulation results. This research will constitute an important milestone on the way toward the implementation of on chip learning algorithms capable of internally programming the synaptic weights in response to input stimuli and/or neural network dynamics.

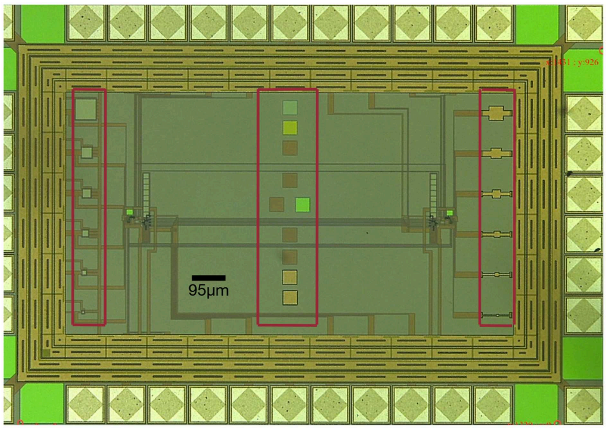


FIGURE 8 | Neuromorphic chip with contact pads for our memristors (red frames). The chip was fabricated using a standard AMS 0,35 μ m CMOS process and supports the integration of memristor devices with neuromorphic synaptic and neural circuits.

6. Conclusion

In summary, we presented several materials as the functional oxide layer in tunnel junction type memristive systems. Magnesia, barium titanate and tantalum oxide indicate that these type of memristive systems can be based on many different materials. A 16 Mb magnetic random access memory is commercially available, demonstrating the good scalability of MTJs that are very similar devices.

As an example, we looked into the analogs of long-term potentiation, long-term depression and STDP in MgO based tunnel junctions. However, the maximum resistance change of up to 8% limits the applicability as synaptic weights in neuromorphic circuits. Here, we increased this resistance change by a factor of 10 in Ta-O based devices, enabling the implementation on top of a neuromorphic chip in the future.

We keep this in mind, but look into the development of autonomous neuromorphic systems now. One of the main obstacles that hindered the development of these systems is the lack of a reliable, robust, and simple implementation of a learning mechanism supported by a low-power compact device suitable for analog storage of the synaptic strength. Key requirements for learning circuits include the long time scale storage capabilities necessary to maintain acquired memories as well as mechanisms for fast modifications of synaptic weights required for the acquisition of new memories. The high integration required in large scale systems poses a severe space constraint impossible to meet with the use of digital memories and associated digital-to-analog converters.

Capacitive storage of synaptic weights has been proposed as a possible solution to this problem. A major drawback of this approach is the overhead required to compensate leakage currents affecting the charge on the capacitor. Two techniques have been proposed in this domain: reduced analog depth of the synaptic weight on long time scales (Fusi et al., 2000; Chicca et al., 2003; Giulioni et al., 2009) and switched capacitors (Vogelstein et al., 2007; Folowosele et al., 2009; Noack et al., 2015). The first solution requires redundancy and therefore large number of synapses as well as power expensive active refresh mechanisms. The second solution requires high frequency digital signals which could introduce deviations in the analog signals due to cross-talk.

Another solution that has been proposed involves the use of floating gates for synaptic strength storage (Holler et al., 1989; Diorio et al., 1998), and for the implementation of Hebbian learning (Gordon and Hasler, 2002) and STDP rules (Ramakrishnan et al., 2011; Nease et al., 2013). Floating gates synapses are suitable for large scale integration but they have the drawback of high voltages required to set the synaptic weight.

Memristors represent an emerging alternative approach to synaptic weight storage thanks to their multi-bit precision storage capability, low energy requirements for writing and nanoscale size. Therefore, we propose the integration of memristive devices in neuromorphic circuits. With this work we are laying the foundations for a possible solution to this ambitious technological challenge.

Acknowledgments

This work was supported by the Ministry of Innovation, Science and Research (MIWF) of North Rhine-Westphalia with an independent researcher grant and by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG). We acknowledge support for the Article Processing Charge by the Deutsche Forschungsgemeinschaft and the Open Access Publication Fund of Bielefeld University. The authors would like to thank Stephen Nease for helpful discussions.

References

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

Borghetti, J., Li, Z., Straznicky, J., Li, X., Ohlberg, D. A. A., Wu, W., et al. (2009). A hybrid nanomemristor/transistor logic circuit capable of self-programming. *Proc. Natl. Acad. Sci. U.S.A.* 106, 1699–1703. doi: 10.1073/pnas.0806642106

Borghetti, J., Snider, G. S., Kuekes, P. J., Yang, J. J., Stewart, D. R., and Williams, R. S. (2010). 'Memristive' switches enable 'stateful' logic operations via material implication. *Nature* 464, 873–876. doi: 10.1038/nature08940

Brader, J. M., Senn, W., and Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Brinkman, W. F., Dynes, R. C., and Rowell, J. M. (1970). Tunneling conductance of asymmetrical barriers. *J. Appl. Phys.* 41, 1915–1921.

Chanthbouala, A., Garcia, V., Cherifi, R. O., Bouzehouane, K., Fusil, S., Moya, X., et al. (2012). A ferroelectric memristor. *Nat. Mater.* 11, 860–864. doi: 10.1038/nmat3415

Chen, X., Wu, G., and Dinghua, B. (2008). Resistive switching behavior of Pt/Mg0.2Zn0.8O/Pt devices for nonvolatile memory applications. *Appl. Phys. Lett.* 93:093501. doi: 10.1063/1.2978158

Chicca, E., Badoni, D., Dante, V., D'Andreagiovanni, M., Salina, G., Carota, L., et al. (2003). A VLSI recurrent network of integrateandfire neurons connected by plastic synapses with long term memory. *IEEE Trans. Neural Netw.* 14, 1297–1307. doi: 10.1109/TNN.2003.816367

Chicca, E., Stefanini, F., Bartolozzi, C., and Indiveri, G. (2014). Neuromorphic electronic circuits for building autonomous cognitive systems. *Proc. IEEE* 102, 1367–1388. doi: 10.1109/JPROC.2014.2313954

Chua, L. (2014). If it's pinched it's a memristor. *Semicond. Sci. Technol.* 29:104001. doi: 10.1088/0268-1242/29/10/104001

Diorio, C., Hasler, P., Minch, B., and Mead, C. (1998). *Three-terminal Silicon Synaptic Device*. U.S. Patent No. 5,825,063. Issued October 20.

Du, X. F., and Chen, I. W. (1998). Fatigue of Pb(Zr0.53Ti0.47)O-3 ferroelectric thin films. *J. Appl. Phys.* 83, 7789–7798.

Eryilmaz, S. B., Kuzum, D., Jeyasingh, R., Kim, S., BrightSky, M., Lam, C., et al. (2014). Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Front Neurosci.* 8:205. doi: 10.3389/fnins.2014.00205

Fiebig, M. (2005). Revival of the magnetoelectric effect. *J. Phys. D Appl. Phys.* 38, 123–152. doi: 10.1088/0022-3727/38/8/R01

Folowosele, F., Etienne-Cummings, R., and Hamilton, T. (2009). "A CMOS switched capacitor implementation of the Mihalas-Niebur neuron," in *Biomedical Circuits and Systems Conference, BIOCAS 2009 (IEEE)* (Beijing), 105–108.

Fusi, S., Annunziato, M., Badoni, D., Salamon, A., and Amit, D. (2000). Spikedriven synaptic plasticity: theory, simulation, VLSI implementation. *Neural Comput.* 12, 2227–2258. doi: 10.1162/089976600300014917

Garcia, V., Fusil, S., Bouzehouane, K., Enouz-Vedrenne, S., Mathur, N. D., Barthélémy, A., et al. (2009). Giant tunnel electroresistance for non-destructive readout of ferroelectric states. *Nature* 460, 81–84. doi: 10.1038/nature08128

Gibbons, J. F., and Beadle, W. E. (1964). Switching properties of thin Nio films. *Solid State Elect.* 7, 785.

Giulioni, M., Pannunzi, M., Badoni, D., Dante, V., and Del Giudice, P. (2009). Classification of overlapping patterns with a configurable analog VLSI neural network of spiking neurons and self-regulating plastic synapses. *Neural Comput.* 21, 3106–3129. doi: 10.1162/neco.2009.08-07-599

Gordon, C., and Hasler, P. (2002). "Biological learning modeled in an adaptive floating-gate system," in *International Symposium on Circuits and Systems, (ISCAS)*, Vol. 5 (Scottsdale, AZ), V609–V612.

Hashimoto, S., Orihara, H., and Ishibashi, Y. (1994). Study of D-E hysteresis loop of TGS based on the Avrami-Type model. *J. Phys. Soc. Jpn.* 63, 1601–1610.

Holler, M., Tam, S., Castro, H., and Benson, R. (1989). An electrically trainable artificial neural network (ETANN) with 10240 floating gate synapses. in *Proceedings of the 1989 IEEE INNS International Joint Conference on Neural Networks*, Vol. 2. (Washington, DC), 191–196.

Huai, Y., Albert, F., Nguyen, P., Pakala, M., and Valet, T. (2004). Observation of spin-transfer switching in deep submicron-sized and low-resistance magnetic tunnel junctions. *Appl. Phys. Lett.* 84, 3118. doi: 10.1063/1.1707228

Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., and Prodromakis, T. (2013). Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology* 24:384010. doi: 10.1088/0957-4484/24/38/384010

Ishibashi, Y., and Takagi, Y. (1971). Note on ferroelectric domain switching. *J. Phys. Soc. Jpn.* 31, 506–510.

Jo, J., Yang, S., Kim, T., Lee, H., Yoon, J, G., Park, S., et al. (2009). Nonlinear dynamics of domain-wall propagation in epitaxial ferroelectric thin films. *Phys. Rev. Lett* 102:045701. doi: 10.1103/PhysRevLett.102.045701

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 1297–1301. doi: 10.1021/nl904092h

Julliere, M. (1975). Tunneling between ferromagnetic films. *Phys. Lett. A*. 54, 225–226. doi: 10.1016/0375-9601(75)90174-7

Kim, S., Hishita, S., Kang, Y. M., and Baik, S. (1995). Structural characterization of epitaxial BaTiO3 thin-films grown by Sputter-Deposition on Mgo(100). *J. Appl. Phys.* 78, 5604–5608.

Kishi, H., Yoda, H., Kai, T., Nagase, T., Kitagawa, E., Yoshikawa, M., et al. (2008). "Lower-current and fast switching of a perpendicular TMR for high speed and high density spin-transfer-torque MRAM," in *IEEE International Electron Devices Meeting, IEDM* (San Francisco, CA), 1–4.

Kozicki, M. N., and Mitkova, M. (2008). *Nanotechnology Vol. 3.* Weinheim: Wiley-VCH.

Krzysteczko, P., Kou, X., Rott, K., Thomas, A., and Reiss, G. (2008). Current induced resistance change of magnetic tunnel junctions with ultra-thin MgO tunnel barriers. *J. Magn. Magn. Mater.* 321, 144. doi: 10.1016/j.jmmm.2008.08.088

Krzysteczko, P., Reiss, G., and Thomas, A. (2009). Memristive switching of MgO based magnetic tunnel junctions. *Appl. Phys. Lett.* 95, 112508. doi: 10.1063/1.3224193

Krzysteczko, P., Münchenberger, J., Schäfers, M., Reiss, G., and Thomas, A. (2012). The memristive magnetic tunnel junction as a nanoscopic synapse-neuron system. *Adv. Mater.* 24, 762–766. doi: 10.1002/adma.201103723

Kubota, H., Ando, Y., Miyazaki, T., Reiss, G., Brueckl, H., Schepper, W., et al. (2003). Size dependence of switching field of magnetic tunnel junctions down to 50 nm scale. *J. Appl. Phys.* 94, 2028–2032. doi: 10.1063/1.1588357

Kuzum, D., Jeyasingh, R. G. D., Lee, B., and Wong, H. S. P. (2012). Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* 12, 2179–2186. doi: 10.1021/nl201040y

Lee, M.-J., Lee, C. B., Lee, D., Lee, S. R., Chang, M., Hur, J. H., et al. (2011). A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta2O5x/TaO2x bilayer structures. *Nat. Mater.* 10, 625–630. doi: 10.1038/nmat3070

Linares-Barranco, B., and Serrano-Gotarredona, T. (2009). Memristance can explain Spike-Time-Dependent-Plasticity in Neural Synapses. *Nat. Precedings*. (in press).

Linn, E., Rosezin, R., Kuegeler, C., and Waser, R. (2010). Complementary resistive switches for passive nanocrossbar memories. *Nat. Mater.* 9, 403–406. doi: 10.1038/nmat2748

Mead, C. (1989). *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley.

Mead, C. (1990). Neuromorphic electronic systems. *Proc. IEEE* 78, 1629–1636.

Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Circ. Syst* 3, 32–42. doi: 10.1109/TBCAS.2008.2005781

Moodera, J., and Mathon, G. (1999). Spin polarized tunneling in ferromagnetic junctions. *J. Magn. Magn. Mater.* 200, 248–273.

Muenchenberger, J., Krzysteczko, P., Reiss, G., and Thomas, A. (2011). Improved reliability of magnetic field programmable gate arrays through the use of memristive tunnel junctions. *J. Appl. Phys.* 110, 096105. doi: 10.1063/1.3660521

Nease, S., Brink, S., and Hasler, P. (2013). "STDP-enabled learning on a reconfigurable neuromorphic platform," in *2013 European Conference on (IEEE) Circuit Theory and Design (ECCTD)* (Dresden), 1–4.

Noack, M., Partzsch, J., Mayr, C. G., Hänzsche, S., Scholze, S., Höppner, S., et al. (2015). Switched-capacitor realization of presynaptic short-term plasticity and stop-learning synapses in 28 nm CMOS. *Front. Neurosci.* 9:10. doi: 10.3389/fnins.2015.00010

Pan, F., Gao, S., Chen, C., Song, C., and Zeng, F. (2014). Recent progress in resistive random access memories: materials, switching mechanisms, and performance. *Mater. Sci. Eng. R Rep.* 83, 1–59. doi: 10.1016/j.mser.2014.06.002

Park, S. W., and Im, H. B. (1992). Effects of oxidation conditions on the properties of tantalum oxide-films on silicon substrates. *Thin Solid Films* 207, 258–264.

Pershin, Y. V., and Di Ventra, M. (2011). Memory effects in complex materials and nanoscale systems. *Adv. Phys.* 60, 145–227. doi: 10.1080/00018732.2010.544961

Pinnow, C. U., and Mikolajick, T. (2004). Material aspects in emerging nonvolatile memories. *J. Electrochem. Soc.* 151, K13–K19. doi: 10.1149/1.1740785

Prezioso, M., Merrikh-Bayat, F., Hoskins, B. D., Adam, G. C., Likharev, K. K., and Strukov, D. B. (2015). Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* 521, 61–64. doi: 10.1038/nature14441

Ramakrishnan, S., Hasler, P., and Gordon, C. (2011). Floating gate synapses with spike-time-dependent plasticity. *IEEE Trans. Biomed. Circuits Syst.* 5, 244–252. doi: 10.1109/TBCAS.2011.2109000

Rose, G., and Dunwiddie, T. V. (1986). Induction of hippocampal long-term potentiation using physiologically patterned stimulation. *Neurosci. Lett.* 69, 244–248.

Rottländer, P., Hehn, M., Lenoble, O., and Schuhl, A. (2001). Tantalum oxide as an alternative low height tunnel barrier in magnetic junctions. *Appl. Phys. Lett.* 78, 3274–3276. doi: 10.1063/1.1374223

Schaefers, M., Drewello, V., Reiss, G., Thomas, A., Thiel, K., Eilers, G., et al. (2009). Electric breakdown in ultrathin MgO tunnel barrier junctions for spin-transfer torque switching. *Appl. Phys. Lett.* 95, 232119. doi: 10.1063/1.3272268

Simpson, R. E., Fons, P., Kolobov, A. V., Fukaya, T., Krbal, M., Yagi, T., et al. (2011). Interfacial phase-change memory. *Nat. Nanotechnol.* 6, 501–505. doi: 10.1038/nnano.2011.96

Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *Nature* 453, 80–83. doi: 10.1038/nature06932

Tagantsev, A. K., Stolichnov, I., Setter, N., Cross, J. S., and Tsukada, M. (2002). Non-Kolmogorov-Avrami switching kinetics in ferroelectric thin films. *Phys. Rev. B.* 66:214109. doi: 10.1103/PhysRevB.66.214109

Thomas, A., Brückl, H., Sacher, M. D., Schmalhorst, J., and Reiss, G. (2003). Aluminum oxidation by a remote electron cyclotron resonance plasma in magnetic tunnel junctions. *J. Vac. Sci. Technol. B* 21, 2120–2122. doi: 10.1116/1.1609480

Thomas, A., Drewello, V., Schaefers, M., Weddemann, A., Reiss, G., Eilers, G., et al. (2008). Direct imaging of the structural change generated by dielectric breakdown in MgO based magnetic tunnel junctions. *Appl. Phys. Lett.* 93, 152508. doi: 10.1063/1.3001934

Thomas, A. (2013). Memristor-based neural networks. *J. Phys. D Appl. Phys.* 46:093001. doi: 10.1088/0022-3727/46/9/093001

Torrezan, A. C., Strachan, J. P., Medeiros-Ribeiro, G., and Williams, R. S. (2011). Sub-nanosecond switching of a tantalum oxide memristor. *Nanotechnology* 22:485203. doi: 10.1088/0957-4484/22/48/485203

Tsymbal, E. Y. (2006). Applied physics: tunneling across a ferroelectric. *Science* 313, 181–183. doi: 10.1126/science.1126230

Valov, I., Waser, R., Jameson, J. R., and Kozicki, M. N. (2011). Electrochemical metallization memories-fundamentals, applications, prospects. *Nanotechnology* 22:289502. doi: 10.1088/0957-4484/22/28/289502

Vogelstein, R., Mallik, U., Vogelstein, J., and Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw.* 18, 253–265. doi: 10.1109/TNN.2006.883007

Von Hippel, A. (1950). Ferroelectricity, domain structure, and phase transitions of barium titanate. *Rev. Mod. Phys.* 22:221. doi: 10.1103/RevModPhys.22.221

Waser, R., and Aono, M. (2007). Nanoionics-based resistive switching memories. *Nat. Mater.* 6, 833–840. doi: 10.1038/nmat2023

Waser, R., Dittmann, R., Staikov, G., and Szot, K. (2009). Redox-based resistive switching memories - nanoionic mechanisms, prospects, and challenges. *Adv. Mater.* 21, 2632–2663. doi: 10.1002/adma.200900375

Waser, R., (ed.). (2008). *Nanotechnology Vol. 3.* Weinheim: Wiley-VCH.

Xia, Q., Robinett, W., Cumbie, M. W., Banerjee, N., Cardinali, T. J., Yang, J. J., et al. (2009). Memristor-CMOS hybrid integrated circuits for reconfigurable logic. *Nano Lett.* 9, 3640–3645. doi: 10.1021/nl901874j

Yan, H., Choe, H., Nam, S., Hu, Y., and Das, S. (2011). Programmable nanowire circuits for nanoprocessors. *Nature* 470, 240–244. doi: 10.1038/nature09749

Yang, J. J., Zhang, M. X., Strachan, J. P., Miao, F., Pickett, M. D., Kelley, R. D., et al. (2010). High switching endurance in TaO[sub x] memristive devices. *Appl. Phys. Lett.* 97, 232102. doi: 10.1063/1.3524521

You, T., Shuai, Y., Luo, W., Du, N., Buerger, D., Skorupa, I., et al. (2014). Exploiting memristive BiFeO3 bilayer structures for compact sequential logics. *Adv. Funct. Mater.* 24, 3357–3365. doi: 10.1002/adfm.201303365

Yun, J.-B., Kim, S., Seo, S., Lee, M.-J., Kim, D.-C., Ahn, S.-E., et al. (2007). Random and localized resistive switching observation in Pt/NiO/Pt. *Phys. Status Solidi Rapid Res. Lett.* 1, 280–282. doi: 10.1002/pssr.200701205

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ORIGINAL RESEARCH ARTICLE
published: 15 January 2015
doi: 10.3389/fnins.2014.00438

A 2-transistor/1-resistor artificial synapse capable of communication and stochastic learning in neuromorphic systems

Zhongqiang Wang, Stefano Ambrogio, Simone Balatti and Daniele Ielmini*

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano and IU.NET, Milano, Italy

Edited by:
Christian G. Mayr, University of Zurich and ETH Zurich, Switzerland

Reviewed by:
Volkan Ozguz, Sabanci University Nanotechnology Research and Application Center, Turkey
Doo Seok Jeong, Korea Institute of Science and Technology, Korea

****Correspondence:***
Daniele Ielmini, Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano and IU.NET, Piazza L. da Vinci 32, 20133 Milano, Italy
e-mail: daniele.ielmini@polimi.it

Resistive (or memristive) switching devices based on metal oxides find applications in memory, logic and neuromorphic computing systems. Their small area, low power operation, and high functionality meet the challenges of brain-inspired computing aiming at achieving a huge density of active connections (synapses) with low operation power. This work presents a new artificial synapse scheme, consisting of a memristive switch connected to 2 transistors responsible for gating the communication and learning operations. Spike timing dependent plasticity (STDP) is achieved through appropriate shaping of the pre-synaptic and the post synaptic spikes. Experiments with integrated artificial synapses demonstrate STDP with stochastic behavior due to (i) the natural variability of set/reset processes in the nanoscale switch, and (ii) the different response of the switch to a given stimulus depending on the initial state. Experimental results are confirmed by model-based simulations of the memristive switching. Finally, system-level simulations of a 2-layer neural network and a simplified STDP model show random learning and recognition of patterns.

Keywords: neuromorphic circuits, spike timing dependent plasticity, neural network, memristor, pattern recognition, cognitive computing

INTRODUCTION

Brain-inspired computing is among the top challenges of the today's information and communication technology. The brain is capable of formidable tasks, such as learning, recognition of visual/auditory patterns, and adaptation in response to new information. To meet this grand challenge, a neuromorphic system should include a number of neurons and synapses similar to a biological human brain, featuring around 10^{12} neurons and 10^{15} synapses (Rajendran et al., 2013). Clearly, such a complex system can be realized only through advanced manufacturing techniques (e.g., 3D integration), and small circuit blocks for neurons and synapses. The latter, in particular, represents by far the largest area of the neuromorphic circuit due to the huge number of inter-neural connections, therefore scaling down the size and complexity of the artificial synapse is a key task in the design of a neuromorphic circuit.

To this purpose, nanoscale resistive switches, or memristors, have been proposed as novel artificial synapses in neuromorphic systems (Likharev et al., 2003; Snider, 2008; Jo et al., 2010). Memristors have the capability of an inherent analog tuning, combined with a 2-terminal structure and a scalable device area and power, therefore they display strong advantage with respect to silicon-based synapses, such as floating gate memories (Diorio et al., 1996) and static RAM (Indiveri et al., 2006). Different switch technologies have been proposed for artificial synapses, including phase change memories (Wright et al., 2011; Bichler et al., 2012; Kuzum et al., 2012), organic-based switches (Bichler et al., 2010), chalcogenide-based switches (Ohno et al., 2011; Suri et al., 2013) and oxide-based resistive switching memories (Seo et al., 2011; Yu et al., 2011, 2013; Park et al., 2012; Ambrogio et al., 2013). The latter approach provides analog switching, nonvolatile behavior, CMOS compatible materials, back-end process and scalable power consumption thanks to filamentary switching (Wong et al., 2012). A memristor naturally satisfies the requisites for electrically-tunable conductance, serving as a connection for communication between a pre-synaptic neuron (PRE) and a post-synaptic neuron (POST), and responsive to the individual spikes fed from both neurons. To achieve this multitask operation, a time-division multiplexing (TDM) approach was previously proposed, where neuron spikes obey a precise synchronous sequence for communication, long-term potentiation (LTP) and long-term depression (LTD) (Snider, 2008; Jo et al., 2010). The synchronous approach, however, may be too idealized with respect to the biological brain, where synapses are potentiated/depressed through asynchronous spike timing dependent plasticity (STDP) (Bi and Poo, 1998). Also, synchronous clocking may be practically difficult in the case of large neuromorphic systems (Zamarreño-Ramos et al., 2011). More recently, a fully asynchronous approach for communication/learning of neuromorphic synapses with leaky-integrate-and-fire (LIF) neurons was proposed (Zamarreño-Ramos et al., 2011; Serrano-Gotarredona et al., 2013). However, a conceptual demonstration of realistic memristor synapses for communication and learning has not been achieved so far.

This work addresses the integration of memristors in neuromorphic systems by introducing a 2-transistor/1-resistor (2T1R) synapse for large scale neuromorphic systems. The transistors in the synapse block allow for (i) multiple-input control of the synapse, which must receive signals from both the PRE and the POST, and (ii) accurate control of the filament growth for analog switching and STDP behavior (Yu et al., 2011; Ambrogio et al., 2013; Subramaniam et al., 2013). STDP in the 2T1R synapse is experimentally demonstrated on bipolar resistive switching memories based on HfO_2 acting as memristive switches. We show that the memristive synapse is capable of communication of spiking signals between neurons and stochastic STDP due to both the natural switching variability in the switch, and to the variations of memristive response depending on the initial state. We finally show a conceptual demonstration of a simulated 2-layer neuromorphic network displaying stochastic pattern learning and recognition, thus further supporting memristive synapse as a scalable, high-functionality building blocks for large scale neuromorphic systems.

MATERIALS AND METHODS

Figure 1A shows the conceptual scheme of the 2T1R structure for the memristive synapse. Both MOS transistors in the synapse control the current flowing through the memristor, thus enabling communication and plasticity. The PRE controls 2 of the 4 terminals of the 2T1R structure, namely the top electrode (TE) and the communication gate (CG). The bottom electrode (BE) is instead connected to the virtual-ground input of the POST, which also controls the fire-gate (FG) terminal.

COMMUNICATION MODE

The usual operation of the synapse consists of the communication mode, where the synapse is a simple resistor with fixed conductance allowing for the weighted transmission of spikes from the PRE to the POST (Zamarreño-Ramos et al., 2011; Indiveri et al., 2013). **Figure 1B** shows the waveforms of the pulses applied to the TE and the CG. The TE pulse includes an exponentially-increasing negative pulse and a short positive pulse, while the CG pulse is a short positive pulse enabling the transmission of a negative current pulse to the POST input through the BE connection. Although the CG voltage is high, it always overlaps with the low-voltage region of the V_{TE} pulse, which rules out any possible switching in the memristor. The negative current spike is integrated by the input stage of the POST as shown in **Figure 1C**, illustrating a single PRE/synapse/POST layer of the neuromorphic network. The integrate-and-fire structure of the POST in **Figure 1C** is largely simplified, in that it does not include, e.g., the leakage path for the stored charge, the refractory period to deactivate integration during fire, and the reset switch to initialize integration after fire (Zamarreño-Ramos et al., 2011). As the PRE spikes collected at the neuron input are integrated, the internal voltage V_{int} increases, eventually hitting the threshold of the comparator stage. This event triggers the fire circuit, namely a monostable circuit delivering spikes in the forward direction, i.e., to the TE and CG terminals of the output synapse, and in the reverse direction, i.e., to the FG terminal of the input synapses.

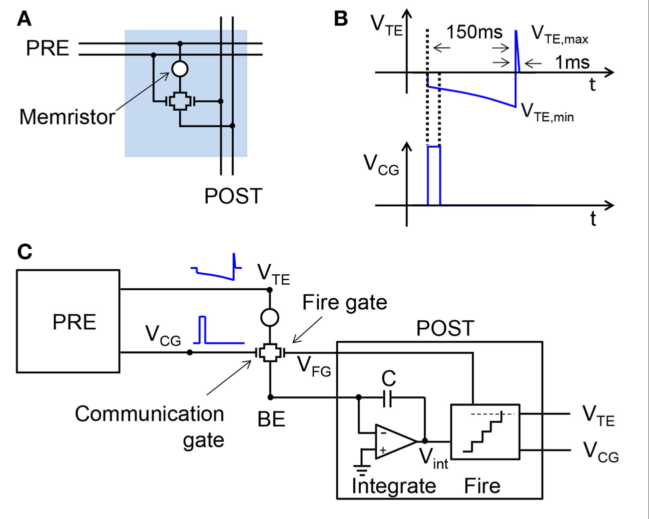


FIGURE 1 | Illustrative scheme for the 2T1R synapse and its operation. The synapse consists of a memristor with 2 series transistors, connected to both the PRE and POST **(A)**. During communication, the PRE delivers pulses to both the CG and the TE terminals of the synapse **(B)**. The resulting current is a function of the memristor conductance and is fed into the input node of the integrate-and-fire POST neuron **(C)**. The maximum and minimum voltages of TE pulse are $V_{TE,max} = 2.4$ V and $V_{TE,min} = -1.65$ V, respectively.

STDP

The temporal coincidence of the PRE spike at the TE of a synapse and of the POST spike (or fire) at the FG of a synapse leads to a change of the memristor conductance according to **Figure 2**. Two cases can be distinguished by the relative delay Δt defined as the time between the end of the negative TE pulse and the end of the FG pulse. For $\Delta t > 0$ in **Figure 2A**, there is an overlap between the positive 1-ms TE pulse and the FG pulse, thus inducing set transition in the memristor. The increase of conductance, due to the growth of a conductive filament (CF) across the HfO_2 switching layer (Nardi et al., 2012), is dictated by the compliance current I_C flowing in the transistor, hence by the gate voltage V_{FG}. Since the FG voltage V_{FG} decreases at increasing Δt, LTP decreases as Δt increases, thus realizing a timing-dependent LTP. **Figure 2B** also includes triangular read pulses at V_{TE} before and after the PRE and POST spikes, both having 1 ms width and a small amplitude of 0.5 V to avoid any disturb to the memristor device. A rectangular pulse of width 1 ms and amplitude 5 V was applied to V_{FG} to enable the pulse operation. The response current during the read pulse before and after the PRE/POST spikes allows to evaluate the increase of conductance induced by LTP.

Similarly, for $\Delta t < 0$ (**Figure 2B**), the negative TE pulse and the positive FG pulse overlap each other, thus inducing reset transition due to the disconnection of the CF. The increase of resistance during reset is controlled by the maximum voltage across the memristor (Nardi et al., 2012), hence by the value of V_{TE}. Since V_{TE} decreases in absolute value at increasing Δt, LTD decreases with Δt, thus carrying out time-dependent LTD. The combination of time-dependent LTP and LTD results in STDP functionality.

CIRCUIT IMPLEMENTATION

To verify the conceptual scheme of STDP in **Figure 2**, we applied the waveform in the figure to a 1T1R structure including an HfO_2 memristor in series with a MOS transistor. The MOS transistor

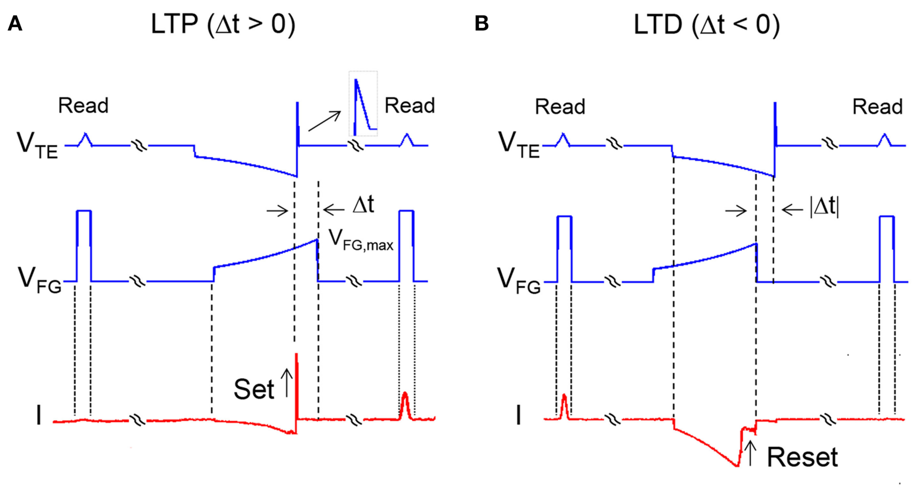


FIGURE 2 | Signal waveforms during LTP and LTD. LTP takes place when the delay Δt between V_{TE} and V_{FG} is positive **(A)**. In this case, there is overlap between the positive 1-ms TE pulse and the FG pulse (maximum voltage 2.9 V), thus inducing set controlled by the V_{FG}-value. V_{FG} increases at decreasing Δt, thus the maximum LTP is obtained for Δt approaching 0. LTD takes place when the delay Δt between V_{TE} and V_{FG} is negative **(B)**. In this case, the negative TE pulse and the positive FG pulse overlap each other, thus inducing reset controlled by the V_{TE}-value. V_{TE} increases in absolute value at decreasing Δt, thus the maximum LTD is obtained for Δt approaching 0.

has threshold voltage $V_T = 1.4$ V, while the channel width and length were 3 μm and 1.425 μm. In the memristor, a Si-doped HfO_2 layer was sandwiched between two TiN electrodes. A Ti cap was deposited between the top TiN electrode and the HfO_2 layer to allow for oxygen extraction aimed at the formation of a local sub-stoichiometric HfO_x ($x < 2$) layer close to the top electrode. This oxygen-exchange layer (OEL) is believed to act as a defect reservoir for the injection during the set transition, when the positive applied voltage induces migration of defects, such as oxygen vacancies and metallic impurities (Hf, Ti) responsible for the formation of a conductive channel, thus resulting in a relatively low resistance. The application of a negative voltage instead results in the retraction of the conductive channel back toward the OEL, thus leading to a relatively high resistance. The HfO_2 layer had an amorphous structure after deposition. The HfO_2 thickness was 10 nm, while the Ti cap thickness was 15 nm. More details about the experimental samples are reported elsewhere (Ambrogio et al., 2014a; Calderoni et al., 2014). The CG transistor was not connected in the experiment, due to our focus on demonstrating STDP. **Figure 1C** shows the conceptual scheme of the 2T1R structure for the memristive synapse. The FG pulse had extreme voltages of 2.9 and 1.0 V, with time constant $\tau = 140$ ms. The same time constant was used for the exponential region of the TE pulse, where the extreme voltages were −1.65 and −0.55 V. The 1-ms half-triangle positive pulse had an extreme amplitude of 2.4 V.

RESULTS

EXPERIMENTAL STDP CHARACTERISTICS

Figure 3A shows the cumulative distributions of measured resistance R in the memristor after application of TE and FG pulses at increasing Δt. The same STDP experiment with a given Δt was repeated 100 times to allow for a sufficient statistical accuracy. The device was always prepared in a full reset state, corresponding to a resistance of about 100 kΩ, and the delay Δt was changed between 1 and 100 ms. The distributions show a decreasing value of R at decreasing delay, in agreement with the expected time-dependent LTP in **Figure 2A**. **Figure 3B** summarizes the conductance enhancement R_0/R, where R_0 is the initial resistance and R is the median value of the distribution. The figure shows time-dependent increase of conductance (LTP) for $\Delta t > 0$, while no change of resistance is obtained for $\Delta t < 0$. **Figure 3C** shows the cumulative distribution of measured R for negative Δt in the range between −1 and −100 ms. To demonstrate LTD, the memristor was initialized in a low resistance state with R_0 around 5 kΩ, obtained with a pulse of 1 ms at $I_C = 170$ μA. **Figure 3D** shows the conductance change R_0/R indicating time-dependent LTD for $\Delta t < 0$. LTD can also be seen at positive delays, which is due to a sequence of reset and set events in the memristor during the negative and positive regions of the TE pulse, respectively. First, a reset transition takes place due to the negative V_{TE}, then the 1-ms V_{TE} pulse induces a set transition with relatively low I_C. As a result, the device is in a set state finally, although with smaller conductance than the initial state, due to the relatively small I_C. Since I_C decreases at increasing positive $\Delta t < 0$, LTD increases with Δt.

Distributions in **Figures 3A,C** show a significant variability, although they were obtained by repeating the same experiments several times on a single device. The distribution variance can be attributed to the switching variability in memristive devices, which was shown to result from the discrete number of defects in the CF (Ambrogio et al., 2014a). The natural switching variability ensures stochastic plasticity in the artificial synapse, where the final state is not deterministically dictated by Δt, rather it is affected by an inherent standard deviation. Note that the relative spread increases with R in **Figure 3**, due to the decreasing number of defects and the correspondingly large statistical spread (Ambrogio et al., 2014a).

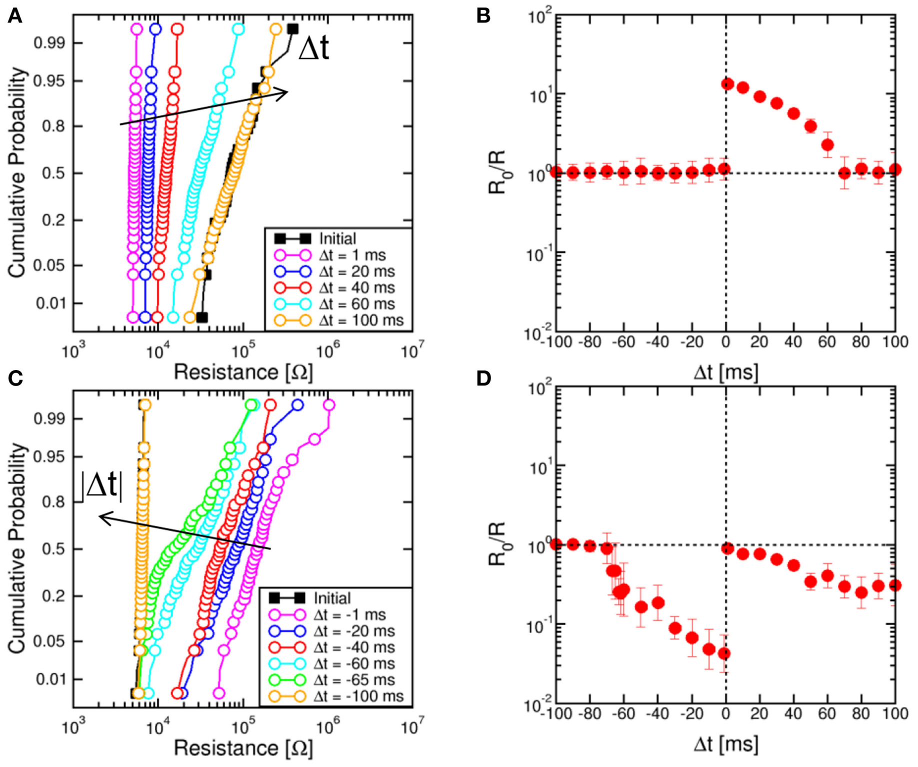


FIGURE 3 | Cumulative distributions of R for variable Δt and corresponding STDP characteristics. Cumulative distributions for $\Delta t > 0$ show an increasing R for increasing Δt, starting from a high-resistance state ($R_0 = 100\,k\Omega$) of the memristor **(A)**. Correspondingly, the conductance change R_0/R decreases at increasing Δt in the STDP characteristic **(B)**. Similarly, for LTD starting from a low-resistance state ($R_0 = 5\,k\Omega$) of the memristor, the cumulative distributions show that R decreases at increasing negative delay **(C)**, while the conductance change R_0/R decreases for large delays in the STDP characteristic **(D)**.

Figure 4 shows STDP characteristics for variable time constant τ in the range between 40 ms and 180 ms, for the memristor initially prepared in a high resistance state (a) or a low resistance state (b). LTP (a) and LTD (b) characteristics show the same behavior as in **Figure 3**, except for a stretching along the Δt axis for increasing τ as a result of the change of the slope of the exponential TE and FG pulses. These results demonstrate the tunability of the STDP characteristics on the timescale through a proper choice of the time constant.

Dependence on the initial state

While results in **Figures 3**, **4** were obtained for the memristor initialized in either the high resistance (for LTP) or the low resistance state (for LTD), it is important to demonstrate the functionality of the STDP scheme for any arbitrary initial state. We first considered variable reset states, obtained by first setting the device to a reference initial low resistance state with a compliance current $I_C = 170\,\mu A$, then resetting the device with a variable maximum negative voltage V_{stop}, as shown in **Figure 5A**. Here, the set and reset transitions in the HfO_2 memristor can be seen at positive and negative voltage, respectively. As the reset voltage increases, the resistance increases, as a result of the increasing growth of the depleted gap along the CF (Nardi et al., 2012). The memristor resistance values were 25, 45, and $100\,k\Omega$ for V_{stop} equal to -1.2, -1.4, and $-1.65\,V$, respectively. Also shown are simulation results according to our physics-based analytical model for resistive switching devices (Ambrogio et al., 2014b). In this model, the Fourier equation for heat generation and conduction is analytically solved, then the local temperature at the injecting point along the CF is used to estimate the migration rate and the corresponding change of CF diameter (during set transition) and depleted gap (during reset transition). The energy barrier controlling ion migration in the analytical model was $E_A = 1.2$ eV. Simulation results in **Figure 5A** support the model as an accurate tool for predicting real memristive switching in metal oxide systems.

Figure 5B shows the measured and calculated STDP characteristics for variable high resistance states in **Figure 5A**. As the initial resistance R_0 increases, the LTP conductance change increases, while the LTD conductance change decreases. However, the shapes of LTP and LTD characteristics are qualitatively the same irrespective of the R_0.

Similarly, we studied variable set state, namely state obtained with variable compliance current during set. **Figure 6A** shows the measured and calculated I–V curves for $I_C = 25$, 50, 100, and $170\,\mu A$. Both set transition at positive voltage and reset transition at negative voltage are shown in the figure. Simulations by the analytical model again accounts closely for the experimental behavior. As I_C increases, the set state resistance decreases, as a result of the larger diameter of the CF (Nardi et al., 2012). Note

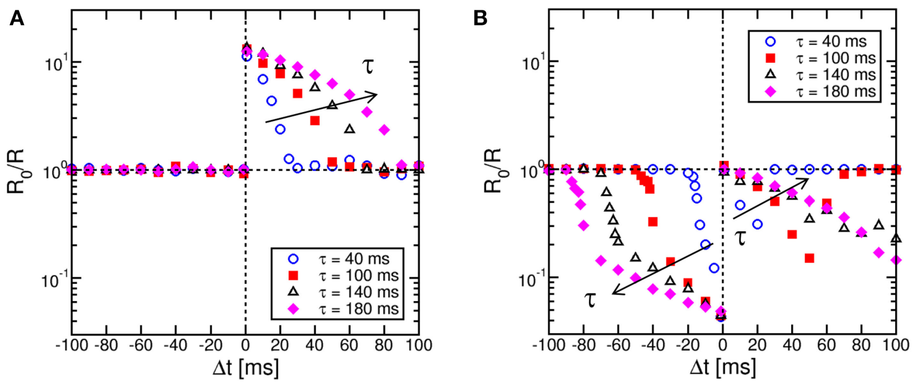


FIGURE 4 | STDP characteristics at increasing time constant τ. The STDP characteristics stretch to longer Δ*t* as the time constant describing the V_{TE} pulse increases, for both LTP on high-resistance states **(A)** and LTD on low-resistance states **(B)**.

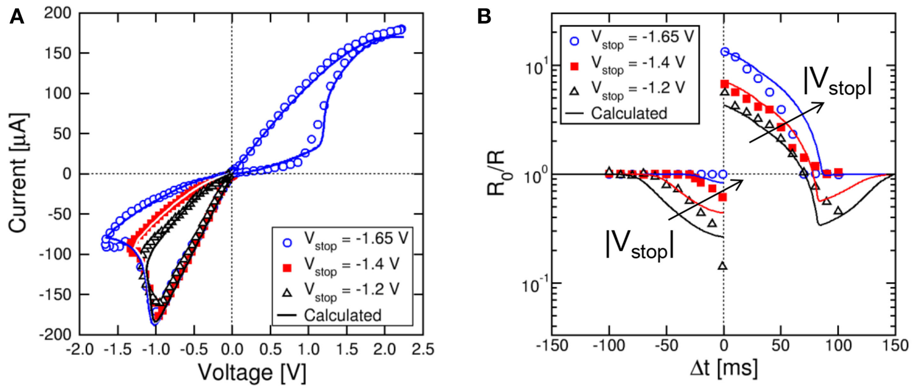


FIGURE 5 | STDP response at variable high-resistance states. Variable high-resistance states are obtained by resetting the memristor device at increasing negative voltage V_{stop} as shown in the I–V curve **(A)**. The STDP characteristics show increasing LTP and decreasing LTD at increasing initial R **(B)**. Analytical calculations well account for the experimental data as a function of V_{stop}.

that the reset current I_{reset} is approximately equal to I_C (Kinoshita et al., 2008; Lee et al., 2008), while the reset voltage V_{reset} is approximately constant around 1 V, marking the voltage needed to initiate defect ionization and migration within the CF (Ielmini, 2011). **Figure 6B** shows the measured and calculated STDP characteristics for variable initial low-resistance state as in **Figure 6A**. Calculations again provide a satisfactory agreement with data and can predict the state-dependent learning in the synapse.

The STDP characteristics in **Figures 5**, **6** show LTD at both positive and negative Δ*t*, which disagrees with the standard timing-dependence of biological learning (Bi and Poo, 1998). However, it was shown that biological synapses might have diversified response based on their function and typologies (Abbott and Nelson, 2000). For instance, a similar STDP response with LTD at positive Δ*t* was observed in hippocampal CA1 neurons (Nishiyama et al., 2000; Wittenberg and Wang, 2006) and explained as due to the Ca^+ dynamics (Caporale and Dan, 2008). This demonstrates that the memristive STDP response in 2T1R synapse is compatible with learning functions in biological neural networks.

Stochastic learning

Results in **Figures 5**, **6** suggests that, for any given Δ*t*, the potentiation/depression of the synapse also depends on the initial state, which introduces a certain degree of stochastic response in the STDP characteristics. To study the stochastic behavior of STDP, we performed experiments with a sequence of coupled TE and FG pulses as in **Figure 2**, applied to the same synapse initially prepared in a high resistance state. A total number of 55 different sequences were applied, each including 10 spikes with randomly chosen Δ*t*. Each random sequence was repeated 50 times to achieve sufficient statistical significance. The time constant was 140 ms in all experiments and simulations.

Figure 7 shows (from top to bottom) the V_{TE} waveform, the V_{FG} waveform and the corresponding resistance R for a random sequence of 10 spikes. Read pulses similar to the waveform in **Figure 2** (not shown in **Figure 7A**) were applied after each spike to measured R. **Figure 7B** shows the color map of the occurrence of any value of conductance change R_0/R as a function of Δ*t* for all 27,500 random spikes. The ratio R_0/R was defined as the ratio between resistances before and after the STDP event. The

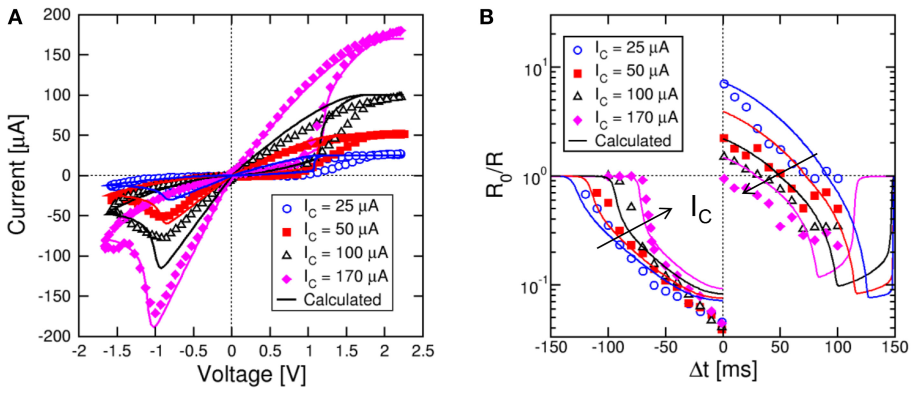


FIGURE 6 | STDP response at variable low-resistance states. Variable low-resistance states are obtained by setting the memristor device at increasing compliance current I_C as shown in the I–V curve **(A)**. The STDP characteristics show increasing LTP at increasing initial R, while LTD characteristics change only slightly **(B)**. Analytical calculations well account for the experimental data as a function of I_C.

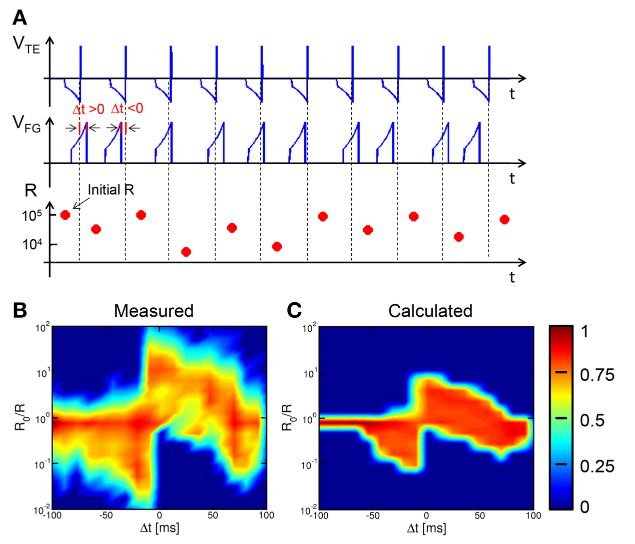


FIGURE 7 | STDP over a random sequence of spikes. A sequence of partially-overlapping PRE/POST spikes with random Δt are applied to the synapse, resulting in LTP or LTD depending on the relative delay **(A)**. The conductance change R_0/R has been collected over 50 repeated experiments with 55 different sequences, each containing 10 random spikes. For any Δt and R_0/R, the probability has been reported in colour scale **(B)**. Calculated results show similar stochastic STDP characteristic **(C)**.

maximum probability (red) indicates LTD for negative Δt and for relatively large positive Δt, while LTP occurs for relatively small positive Δt. **Figure 7C** shows the color map of R_0/R as a function of Δt for 10^4 simulated sequences assuming random Δt and using the same analytical switching model for the memristor as in **Figures 5**, **6**. The calculated color map shows a qualitative agreement with the experimental STDP, indicating potentiation at small $\Delta t > 0$, and depression at negative Δt and large positive Δt. The STDP statistics, where different LTP and/or LTD are obtained for any given Δt, is mainly due to the dependence on the initial state as discussed in Section Experimental STDP Characteristics Experimental data in **Figure 7B** indicate a larger spread of R_0/R, which we attribute to the additional source variability due to the naturally stochastic switching, i.e., the physical origin of the distribution spread in **Figures 3A,C**.

The impact of switching variability is also highlighted in **Figure 8**, showing the values of R measured after each spike in a sequence of 10 events with random timing Δt. **Figure 8A** compares 5 typical sequences always starting from the same initial high resistance state (about 10^5 Ω), to study the effect of switching variability. The measured R displays random walk depending on Δt, which is shown in **Figure 8B**. Note the significant random change among all trajectories due to the stochastic switching during each set/reset operations. The largest variability is seen for LTD, due to the large variability in the high resistance state (see, e.g., **Figures 3A,C**). On the other hand, LTP leads to a certain decrease of variability, since the set operation is mainly controlled by I_C and negligibly depends on the initial high-resistance state (Ambrogio et al., 2014a).

PATTERNING LEARNING AND RECOGNITION THROUGH STDP

To verify that STDP in the 2T1R synapse is capable of pattern learning and recognition, we adopted a 2 layer neuromorphic network schematically shown in **Figure 9**. Here, N pre-synaptic neurons provide spiking input to M post-synaptic neurons through an array of NxM synapses (Zamarreño-Ramos et al., 2011). Connections to PRE and POST in **Figure 9** are organized according to rows and columns, respectively, each requiring 2 lines for connecting the 2T1R synapse, namely the TE and CG line from PRE to the synapse and the BE and the FG between the synapse and the POST.

To simulate pattern learning, we assumed that the N PRE neurons belong to an artificial retina providing visual stimuli corresponding to the 8 × 8 square pattern at the extreme left in **Figure 10A** ($N = 64$). The pattern was fed synchronously from PRE to POST through the synapse array, by applying a spike for every white pixel while black pixel did not yield any spike. The pattern was randomly alternated with random noise, consisting of 95% probability for black and 5% for white signals in each of the N pixels. The duty cycle of true pattern occurrence was

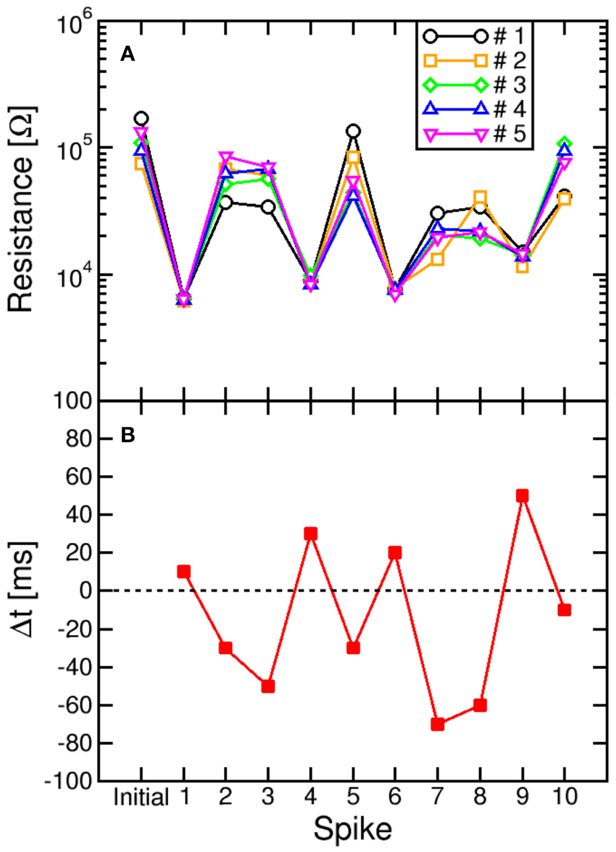


FIGURE 8 | Stochastic LTP and LTD. The resistance **(A)** was plotted as a function of the number of the spike within a sequence with determined delay Δt **(B)**. The sequence was repeated 5 times to highlight the variability of resistance change during LTP and LTD. natural switching variability leads to random walk of R during each set/reset operation, with reset (LTD) process showing generally larger stochastic variation compared to set (LTP) process.

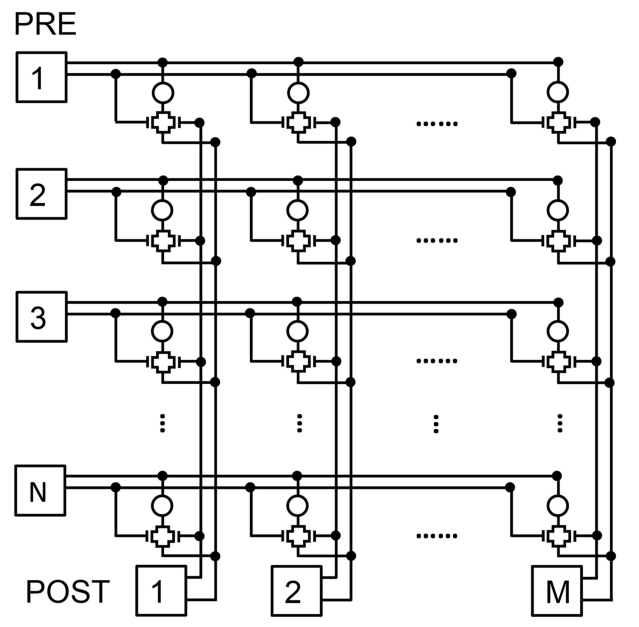


FIGURE 9 | Schematic illustration of the 2-layer neuromorphic network. The first layer consists of N PRE, while the second layer consists of M POST, thus resulting in a network of NxM synapses with 2T1R structure.

50%. All signals received at a POST were integrated according to the scheme in **Figure 1C**, then a fire signal was triggered as the internal potential V_{int} reached a given threshold. The fire signals were delivered from the POST to all connected synapses, and dictated a conductance change according to the simplified STDP characteristic in **Figure 10B**. This includes LTP for small $\Delta t > 0$ and LTD for $\Delta t < 0$ and for large $\Delta t > 0$, according to the most general response of the 2T1R synapse in **Figures 5**, **6**. As a minimum resistance $R = 5$ kΩ was reached, further potentiation was inhibited in the synapse, while depression was inhibited above a resistance $R = 100$ kΩ.

Figure 10C shows the calculated conductance 1/R for 64 synapses in a single column, which connected all PRE to a single POST. Starting from a uniformly distributed random initial state, the synapse conductance, or weight, generally follows 2 trends, either increasing or decreasing with time due to repeated LTP and LTD. The evolution of the synapse weights is also shown in **Figure 10A** for 4 states, namely initial state and after 100, 500, and 1000 epochs of pattern presentation. The pattern is seen to rapidly potentiate the corresponding synapses, with potentiation and depression occurring in white and black pixel positions, respectively. On the other hand, a longer time is needed for depression of unstimulated synapses, since depression relies on uncorrelated random noise patterns. While potentiation of pattern synapses takes about 30 epochs, the depression of other synapses is completed in about 500 epochs. These results fully support the capability for pattern learning and recognition by the scheme in **Figure 2**, combined with the STDP response of our 2T1R synapse which was simplified in **Figure 10B**.

A 2-layer network similar to **Figure 9** was previously shown to lead to random specialization of POST neurons to distinct patterns, such as the cars appearing in specific lanes on the highway (Bichler et al., 2012). We verified the random specialization in our system by considering a NxM network as in **Figure 9** with $N = 64$ (number of pixels in the pattern and number of PRE neurons) and $M = 10{,}000$ (number of POST neurons). We presented the 2 patterns in **Figure 11A** and b in a random sequence of patterns (70% probability equally distributed between pattern 1 and 2) and random noise (30% probability). The initial values of the synapses were randomly distributed as in **Figure 10**. **Figure 11C** shows the percentage distributions of patterns recognized after a total number of 10^3 epochs: Patterns 1 and 2 were recognized with equal probability of about 48%, while no recognition was possible in 4% of the cases. Most of these recognition failure are due to incorrect recognition of the two patterns, converging to a mixture of patterns 1 and 2, while some errors are due to very slow learning, leading to incomplete learning at the final calculated epoch. **Figure 11D** shows the probability distributions for potentiating, hence learning, pattern 1 and 2, identified as the first epoch with all synapses completely potentiated. Both distributions peak at about 20 epochs, with no preference for any of the

2 patterns. Note that the patterns 1 and 2 were selected to have the same number of black/white pixels, to ensure a constant average firing rate of the POST. This accounts for the equal learning times in **Figure 11C**. **Figure 11D** also shows the distribution of times corresponding to the depression of all the synapses not belonging to pattern 1 or 2. The distributions show a similar behavior and peak at 500 epochs. The different timescale is caused by the fact that depression is due to uncorrelated spikes originated by random noise, while pattern learning is linked to the density of patterns 1 or 2 and their related input frequency.

DISCUSSION

The proposed synapse circuit allows for asynchronous transmission and plasticity controlled by the spiking delay between the pre- and post-synaptic neurons. The synapse circuit adheres to the conventional organization of the neural network, where integrate-and-fire neurons serve as both input and output of the communication and plasticity. In particular, the BE terminal, being connected to the virtual ground input of the neuron, serves as reference ground for the synapse circuit, while pulses of arbitrary voltage are applied to the other 3 terminals, namely TE,

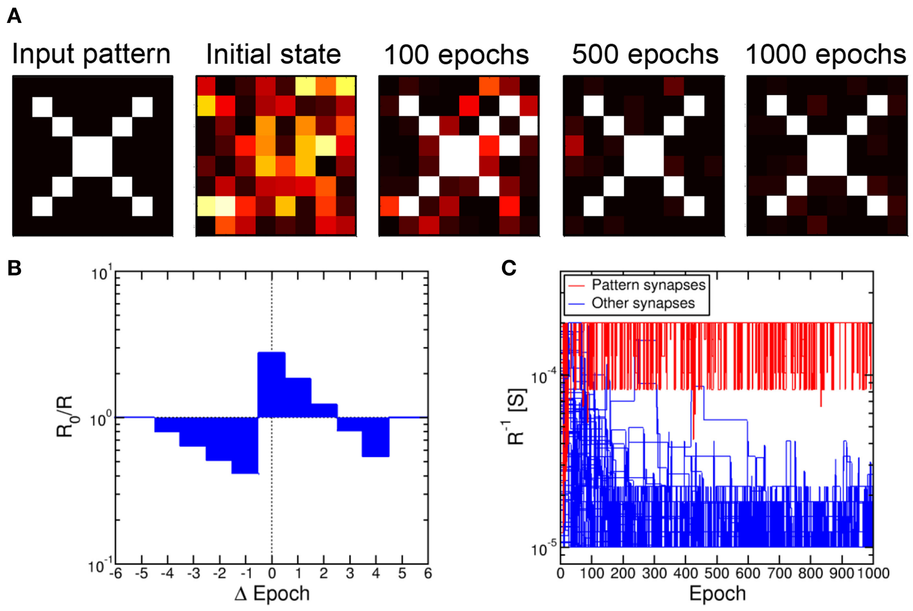


FIGURE 10 | Pattern learning and recognition through 2T1R synapses. The input pattern was fed by the first layer of 8 PRE neurons toward a second layer of 8 POST neurons, resulting in learning as demonstrated by the evolution of the synapse weights **(A)**. Each synapse was changed according to a simplified STDP characteristic with discrete delay **(B)**. The conductance of pattern synapses increases due to the learning process, while other synapses experience increasing depression **(C)**.

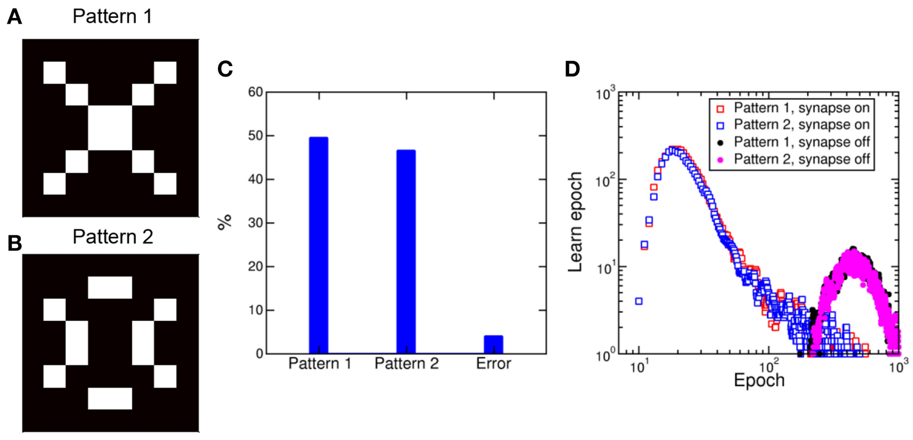


FIGURE 11 | Pattern competition during learning. Random submission of pattern 1 **(A)** and pattern 2 **(B)** in a 8 × 8 synapse array results in learning of either pattern with equal probability approaching 50%, including a minority of error due to transition from one pattern to the other **(C)**. Potentiation of pattern synapses takes place in about 20 epochs, while depression of out-of-pattern synapses requires around 500 epochs **(D)**.

CG and FG. This is different from previous approaches, where the pre-synaptic pulse (spike) and the post-synaptic pulse (fire) where applied to the TE and BE, respectively, of the resistive synapse (Yu et al., 2011; Indiveri et al., 2013). It is also different from other approaches employing 1T1R structures, where STDP relied on a dynamic V_T behavior of the transistor, achieved through nanoparticle-containing gate dielectric (Subramaniam et al., 2013). In fact, only standard transistor CMOS transistor are needed in the 2T1R synapse in this work.

The transistors in the 2T1R structure are functional in achieving 2 necessary behaviors of the synapse array, namely STDP and communication. On the one hand, the FG transistor allows for a spike timing comparison between two pulses, namely the TE pulse from the pre-synaptic neuron and the FG pulse from the post-synaptic neuron (Ambrogio et al., 2013). Therefore, the FG transistor is functional to plasticity. On the other hand, the CG transistor allows for enabling communication from pre-synaptic neuron to post-synaptic neuron in the neural network. If there was no CG transistor, the TE pulse might affect the weight of the synapse even without any fire from the post-synaptic neuron. Note in fact that the CG voltage is high only during the initial part of the TE pulse, at relatively low voltage. Therefore, this transistor is functional to communication, while protecting the memristor from the rather large TE voltage used for plasticity.

In addition, transistors allow to limit the current flowing in the memristive switch during the set transition, thus preventing uncontrolled switching and even irreversible breakdown of the device. These latter events may result in excessive power consumption due to low resistance value in the synapse, and/or in the impossibility to reset the memristor because of excessive growth of the conductive channel. Current limitation can be achieved by biasing the transistor in the saturated regime at relatively low gate voltage, which ensures that the maximum current after set transition is limited. Finally, the transistor serves as selector in the synapse array of **Figure 9**, which otherwise would be plagued by significant sneak-path currents (Baek et al., 2005). Note that other types of selectors would allow better scalability of the array, e.g., p-n diodes (Baek et al., 2005), or threshold switch devices (Cha et al., 2013), thanks to the 2-terminal structure. However, 2 terminals would not be sufficient for the local comparison of spike timing which is needed for synapse plasticity control.

It has been pointed out that the necessity to generate dedicated waveforms within the neuron circuit might lead to an excessive circuit overhead, thus conflicting with the need for very large scale arrays with high synaptic densities (Kornijcuk et al., 2014). Note however that the generator of the spike belongs to the neuron circuit, thus a complex waveform should not affect the density of synapses. Also, note that the waveforms in **Figures 1**, **2** have been designed to achieve a bio-realistic STDP as shown in **Figure 4**. Other waveforms and STDP characteristics can be used with no impact on the pattern recognition capability, while strongly alleviating the burden on the neuron circuit. This is demonstrated in **Figure 12**, showing the square waveforms for V_{TE} and V_{FG} (a) and the corresponding statistical STDP characteristic (b) obtained from 7.5×10^4 random spikes. Note that the STDP characteristics reflects the simple shape of the spike and fire pulses, while we demonstrated that the pattern learning behavior is not affected. This further demonstrates the strength of the STDP process and the flexibility of our 2T1R circuit in realizing LTP and LTD with a variety of spike shapes. Note that pulse widths of the neuron spikes in the range of 100 ms, which are needed to achieve real-time bio-compatible neuromorphic behavior (Indiveri et al., 2011), do not necessarily require large capacitors. In fact, time responses in the 100 ms range are straightforwardly achieved in neuromorphic circuits through relatively small capacitances (e.g., 1 pF) charged/discharged by extremely low current in MOS transistors biased in the subthreshold regime (Mitra et al., 2009).

Low-power operation is a fundamental property of neuromorphic circuits. The energy consumption of our 2T1R synapse for communication can be estimated to about 150 nJ from the voltage waveform in **Figure 1B** assuming $I = 50\,\mu A$. Assuming an average spike frequency of 1 Hz, the power consumption for communication should be around 150 nW. This value can be reduced by decreasing the pulse-width of the V_{CG} pulse and the current during communication. On the other hand, the energy consumption is slightly larger due to the larger voltage and current needed for resistive switching. For instance, the LTP energy is around 400 nJ for a current of 170 µA and a V_{TE} of 2.4 V in

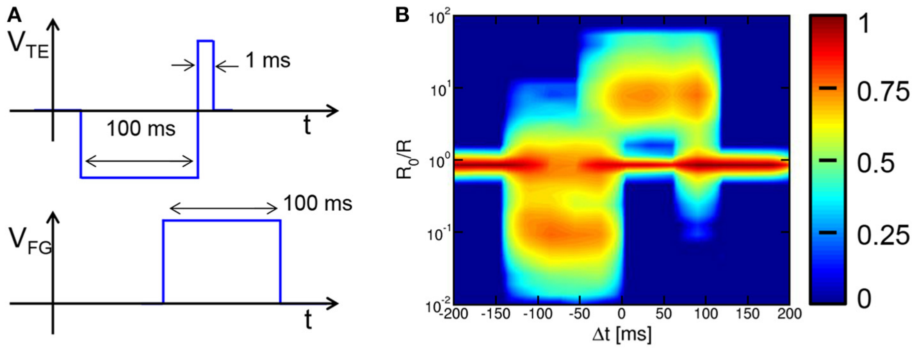


FIGURE 12 | Square-pulse STDP. The use of square pulses for V_{TE} and V_{FG} **(A)** allows to achieve STDP with square characteristics suitable to learning and recognition **(B)**, while requiring a simple circuit for pulse generation.

correspondence of the positive peak. However, since the LTP frequency is expected to be smaller than the spiking frequency, the power consumption for LTP might be in the same range as the communication power. Similar to the communication case, LTP power can be reduced by properly decreasing the current (e.g., by up to a factor 10) and the pulse width (up to a factor 10^3). This allows for memristive-based synapses with relatively low power consumption.

Other switching concepts might be used in alternative to oxide memristors, e.g., spin-transfer-torque (STT) elements (Locatelli et al., 2014) or phase change memory (PCM) elements (Kuzum et al., 2012; Eryilmaz et al., 2014). However, oxide memristors allows for a smaller power consumption since the switching channel area can be controlled through the transistor current during the set transition, whereas the switching current is controlled by the lithography-defined area of the device in both STT and PCM devices, which thus can hardly be reduced below 50 μA (Ielmini and Lacaita, 2011; Kim et al., 2011).

The use of a HfO_2 memristor allows for CMOS compatible process in the back-end, however other metal oxides can be used in principle for the active switching layer, such as TaO_x (Lee et al., 2011). A careful material engineering is needed to identify the best material properties for synaptic functionality, including, e.g., controllability of the synapse weight, stochastic switching and low power operation.

ACKNOWLEDGMENT

We gratefully acknowledge Micron Technology for kindly providing experimental resistive switching samples for this study.

REFERENCES

Abbott, L. F., and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nat. Neurosci.* 3(Suppl), 1178–1183. doi: 10.1038/81453

Ambrogio, S., Balatti, S., Cubeta, A., Calderoni, A., Ramaswamy, N., and Ielmini, D. (2014a). Statistical fluctuations in HfO_x resistive-switching memory (RRAM): Part I – Set/Reset variability. *IEEE Trans. Electron Devices* 61, 2912–2919. doi: 10.1109/TED.2014.2330200

Ambrogio, S., Balatti, S., Gilmer, D. C., and Ielmini, D. (2014b). Analytical modeling of oxide-based bipolar resistive memories and complementary resistive switches. *IEEE Trans. Electron Devices* 61, 2378–2386. doi: 10.1109/TED.2014.2325531

Ambrogio, S., Balatti, S., Nardi, F., Facchinetti, S., and Ielmini, D. (2013). Spike-timing dependent plasticity in a transistor-selected resistive switching memory. *Nanotechnology* 24:384012. doi: 10.1088/0957-4484/24/38/384012

Baek, I. G., Kim, D. C., Lee, M. J., Kim, H.-J., Yim, E. K., Lee, M. S., et al. (2005). Multi-layer cross-point binary oxide resistive memory (OxRRAM) for Post-NAND storage application. *IEDM Tech. Dig.* 750–753. doi: 10.1109/IEDM.2005

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

Bichler, O., Suri, M., Querlioz, D., Vuillaume, D., DeSalvo, B., and Gamrat, C. (2012). Visual pattern extraction using energy-efficient 2-PCM synapse neuromorphic architecture. *IEEE Trans. Electron Devices* 59, 2206–2214. doi: 10.1109/TED.2012.2197951

Bichler, O., Zhao, W., Alibart, F., Pleutin, S., Vuillaume, D., and Gamrat, C. (2010). Functional model of a nanoparticle organic memory transistor for use as a spiking synapse. *IEEE Trans. Electron Devices* 57, 3115–3122. doi: 10.1109/TED.2010.2065951

Calderoni, A., Sills, S., and Ramaswamy, N. (2014). "Performance comparison of O-based and Cu-based ReRAM for high-density applications," in *International Memory Workshop* (Taipei), 1–4.

Caporale, N., and Dan, Y. (2008). Spike timing-dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.* 31, 25–46. doi: 10.1146/annurev.neuro.31.060407.125639

Cha, E., Woo, J., Lee, D., Lee, S., Song, J., Koo, Y., et al. (2013). Nanoscale (∼10nm) 3D vertical ReRAM and NbO_2 threshold selector with TiN electrode. *IEDM Tech. Dig.* 268–271. doi: 10.1109/IEDM.2013.6724602

Diorio, C. J., Hasler, P. E., Mead, C. A., and Minch, B. A. (1996). A single-transistor silicon synapse. *IEEE Trans. Electron Devices* 43, 1972–1980. doi: 10.1109/16.543035

Eryilmaz, S. B., Kuzum, D., Jeyasingh, R., Kim, S., BrightSky, M., Lam, C., et al. (2014). Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Front. Neurosci.* 8:205. doi: 10.3389/fnins.2014.00205

Ielmini, D. (2011). Modeling the universal set/reset characteristics of bipolar RRAM by field- and temperature-driven filament growth. *IEEE Trans. Electron Devices* 58, 4309–4317. doi: 10.1109/TED.2011.2167513

Ielmini, D., and Lacaita, A. L. (2011). Phase change materials in non-volatile storage. *Mater. Today* 14, 600–607. doi: 10.1016/S1369-7021(11)70301-7

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., and Prodromakis, T. (2013). Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology* 24:384010. doi: 10.1088/0957-4484/24/38/384010

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 1297–1301. doi: 10.1021/nl904092h

Kim, W., Jeong, J. H., Kim, Y., Lim, W. C., Kim, J. H., Park, J. H., et al. (2011). Extended scalability of perpendicular STT-MRAM towards sub-20nm MTJ node. *IEDM Tech. Dig.* 531–534. doi: 10.1109/IEDM.2011.6131602

Kinoshita, K., Tsunoda, K., Sato, Y., Noshiro, H., Yagaki, S., Aoki, M., et al. (2008). Reduction in the reset current in a resistive random access memory consisting of NiO_x brought about by reducing a parasitic capacitance. *Appl. Phys. Lett.* 93, 033506. doi: 10.1063/1.2959065

Kornijcuk, V., Kavehei, O., Lim, H., Seok, J. Y., Kim, S. K., Kim, I., et al. (2014). Multiprotocol-induced plasticity in artificial synapses. *Nanoscale* 6, 15151–15160. doi: 10.1039/C4NR03405H

Kuzum, D., Jeyasingh, R. G. D., Lee, B., and Wong, H.-S. P. (2012). Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* 12, 2179–2186. doi: 10.1021/nl201040y

Lee, H. Y., Chen, P. S., Wu, T. Y., Chen, Y. S., Wang, C. C., Tzeng, P. J., et al. (2008). Low power and high speed bipolar switching with a thin reactive Ti buffer layer in robust HfO_2 based RRAM. *IEDM Tech. Dig.* 297–300. doi: 10.1109/IEDM.2008.4796677

Lee, M.-J., Lee, C. B., Lee, D., Lee, S. R., Chang, M., Hur, J. H., et al. (2011). A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures. *Nat. Mater.* 10, 625–630. doi: 10.1038/nmat3070

Likharev, K. K., Mayr, A., Muckra, I., and Türel, Ö. (2003). CrossNets – high-performance neuromorphic architectures for CMOL circuits. *Ann. N.Y. Acad. Sci.* 1006, 146–163. doi: 10.1196/annals.1292.010

Locatelli, N., Cros, V., and Grollier, J. (2014). Spin-torque building blocks. *Nat. Mater.* 13, 11–20. doi: 10.1038/nmat3823

Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Cir. Syst.* 3, 32–43. doi: 10.1109/TBCAS.2008.2005781

Nardi, F., Larentis, S., Balatti, S., Gilmer, D. C., and Ielmini, D. (2012). Resistive switching by voltage-driven ion migration in bipolar RRAM – Part I: experimental study. *IEEE Trans. Electron Devices* 59, 2461–2467. doi: 10.1109/TED.2012.2202319

Nishiyama, M., Hong, K., Mikoshiba, K., Poo, M.-M., and Kato, K. (2000). Calcium stores regulate the polarity and input specificity of synaptic modification. *Nature* 408, 584–588. doi: 10.1038/35022604

Ohno, T., Hasegawa, T., Tsuruoka, T., Terabe, K., Gimzewski, J. K., and Aono, M. (2011). Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. *Nat. Mater.* 10, 591–595. doi: 10.1038/nmat3054

Park, S., Kim, H., Choo, M., Noh, J., Sheri, A., Jung, S., et al. (2012). "RRAM-based synapse for neuromorphic system with pattern recognition function," in *Electron Devices Meeting (IEDM), 2012 IEEE International* (San Francisco, CA). doi: 10.1109/IEDM.2012.6479016

Rajendran, B., Liu, Y., Seo, J.-S., Gopalakrishnan, K., Chang, L., Friedman, D. J., et al. (2013). Specifications of nanoscale devices and circuits for neuromorphic computational systems. *IEEE Trans. Electron Devices* 60, 246–253. doi: 10.1109/TED.2012.2227969

Seo, K., Kim, I., Jung, S., Jo, M., Park, S., Park, J., et al. (2011). Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device. *Nanotechnology* 22:254023. doi: 10.1088/0957-4484/22/25/254023

Serrano-Gotarredona, T., Masquelier, T., Prodromakis, T., Indiveri, G., and Linares-Barranco, B. (2013). STDP and STDP variations with memristors for spiking neuromorphic learning systems. *Front. Neurosci.* 7:2. doi: 10.3389/fnins.2013.00002

Snider, G. S. (2008). "Spike-timing-dependent learning in memristive nanodevices," in *IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH* (Anaheim), 85–92.

Subramaniam, A., Cantley, K. D., Bersuker, G., Gilmer, D. C., and Vogel, E. M. (2013). Spike-timing-dependent plasticity using biologically realistic action potentials and low-temperature materials. *IEEE Trans. Nanotechnol.* 12, 450–454. doi: 10.1109/TNANO.2013.2256366

Suri, M., Querlioz, D., Bichler, O., Palma, G., Vianello, E., Vuillaume, D., et al. (2013). Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Trans. Electron Devices* 60, 2402–2409. doi: 10.1109/TED.2013.2263000

Wittenberg, G. M., and Wang, S. S.-H. (2006). Malleability of spike-timing-dependent plasticity at the CA3–CA1 synapse. *J. Neurosci.* 26, 6610–6617. doi: 10.1523/JNEUROSCI.5388-05.2006

Wong, H.-S. P., Lee, H.-Y., Yu, S., Chen, Y.-S., Wu, Y., Chen, P.-S., et al. (2012). Metal–Oxide RRAM. *Proc. IEEE* 100, 1951–1970. doi: 10.1109/JPROC.2012.2190369

Wright, C. D., Liu, Y., Kohary, K. I., Aziz, M. M., and Hicken, R. J. (2011). Arithmetic and Biologically-inspired computing using phase-change materials. *Adv. Mater.* 23, 3408–3413. doi: 10.1002/adma.201101060

Yu, S., Gao, B., Fang, Z., Yu, H., Kang, J., and Wong, H.-S. P. (2013). A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation. *Adv. Mater.* 25, 1774–1779. doi: 10.1002/adma.201203680

Yu, S., Wu, Y., Jeyasingh, R., Kuzum, D., and Wong, H.-S. P. (2011). An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation. *IEEE Trans. Electron Devices* 58, 2729–2737. doi: 10.1109/TED.2011.2147791

Zamarreño-Ramos, C., Camuñas-Mesa, L. A., Pérez-Carrasco, J. A., Masquelier, T., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex. *Front. Neurosci.* 5:26. doi: 10.3389/fnins.2011.00026

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 21 October 2014; accepted: 12 December 2014; published online: 15 January 2015.

Citation: Wang Z, Ambrogio S, Balatti S and Ielmini D (2015) A 2-transistor/1-resistor artificial synapse capable of communication and stochastic learning in neuromorphic systems. Front. Neurosci. ***8****:438. doi: 10.3389/fnins.2014.00438*

This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 16 December 2014
doi: 10.3389/fnins.2014.00412

A compound memristive synapse model for statistical learning through STDP in spiking neural networks

Johannes Bill* and Robert Legenstein

Faculty of Computer Science and Biomedical Engineering, Institute for Theoretical Computer Science, University of Technology, Graz, Austria

Edited by:
Christian G. Mayr, University of Zurich and ETH Zurich, Switzerland

Reviewed by:
Nabil Imam, Cornell University, USA
Love Mattias Cederström, Technische Universität Dresden, Germany

****Correspondence:***
Johannes Bill, Institute for Theoretical Computer Science, University of Technology, Inffeldgasse 16b/1, A-8010 Graz, Austria
e-mail: bill@tugraz.at

Memristors have recently emerged as promising circuit elements to mimic the function of biological synapses in neuromorphic computing. The fabrication of reliable nanoscale memristive synapses, that feature continuous conductance changes based on the timing of pre- and postsynaptic spikes, has however turned out to be challenging. In this article, we propose an alternative approach, the compound memristive synapse, that circumvents this problem by the use of memristors with binary memristive states. A compound memristive synapse employs multiple bistable memristors in parallel to jointly form one synapse, thereby providing a spectrum of synaptic efficacies. We investigate the computational implications of synaptic plasticity in the compound synapse by integrating the recently observed phenomenon of stochastic filament formation into an abstract model of stochastic switching. Using this abstract model, we first show how standard pulsing schemes give rise to spike-timing dependent plasticity (STDP) with a stabilizing weight dependence in compound synapses. In a next step, we study unsupervised learning with compound synapses in networks of spiking neurons organized in a winner-take-all architecture. Our theoretical analysis reveals that compound-synapse STDP implements generalized Expectation-Maximization in the spiking network. Specifically, the emergent synapse configuration represents the most salient features of the input distribution in a Mixture-of-Gaussians generative model. Furthermore, the network's spike response to spiking input streams approximates a well-defined Bayesian posterior distribution. We show in computer simulations how such networks learn to represent high-dimensional distributions over images of handwritten digits with high fidelity even in presence of substantial device variations and under severe noise conditions. Therefore, the compound memristive synapse may provide a synaptic design principle for future neuromorphic architectures.

Keywords: neuromorphic, synapse, synaptic plasticity, STDP, memristor, WTA, Bayesian inference, unsupervised learning

1. INTRODUCTION

A characteristic property of massively parallel computation in biological and artificial neural circuits is the need for intensive communication between neuronal elements. As a consequence, area- and energy consumption of neuromorphic circuits is often dominated by those circuits that implement synaptic transmission between neural elements (Schemmel et al., 2008). Biological synapses are highly dynamic computational entities, exhibiting plasticity on various time scales (Malenka and Bear, 2004). In order to capture the most salient of these aspects, silicon synapses thus demand extensive circuitry if implemented in CMOS technology. On this account, novel nanoscale circuit elements have recently gained interest in the field of neuromorphic engineering as a promising alternative solution for the implementation of artificial synaptic connections. In particular, for mixed-signal neuromorphic CMOS architectures, which combine traditional digital circuits with analog components, memristors are considered a promising class of circuit elements due to high integration densities, synapse-like plasticity dynamics, and low power consumption (Choi et al., 2009; Jo et al., 2010; Kuzum et al., 2011; Indiveri et al., 2013). One particularly important feature of memristors is that their electrical resistance (often termed "memristance") can be altered in a persistent manner by applying a voltage to its terminals, leading to the eponymous perception of memristors as resistors with memory. As an important application of this property, it was shown that the memristance can be changed based on the spike timings of the pre- and postsynaptic neurons in a manner that approximates spike-timing dependent plasticity (STDP), a plasticity rule that is believed to represent a first approximation for the changes of synaptic efficacies in biological synapses (Markram et al., 1997; Caporale and Dan, 2008; Markram et al., 2012). On the level of single synapses, this important property has been confirmed experimentally in real memristors (Mayr et al., 2012) and has been included into computational models of memristive plasticity (Serrano-Gotarredona et al., 2013). On the network level, models of memristive STDP were employed in computer simulations to demonstrate the potential applicability of neuromorphic designs with memristive synapses to pattern recognition tasks (Querlioz et al., 2011).

In practice however, the production of functional memristive synapses with nanoscale dimensions has proven difficult, mainly due to large device variations and their unreliable behavior. It turned out to be particularly challenging to fabricate reliable nanoscale memristive synapses that feature a continuous spectrum of conductance values. As an alternative solution, it was proposed to employ bistable memristors as neuromorphic synapses instead since they exhibit a high degree of uniformity (Fang et al., 2011) and high durability (Jo et al., 2009b). For switching between their two stable conductance states, bistable memristors can be operated in a deterministic as well as in a stochastic regime (Jo et al., 2009b).

Using machine learning theory, we show in this article that stochastically switching bistable memristors become computationally particularly powerful in mixed-signal neuromorphic architectures when multiple memristors are combined to jointly form one synapse. Such a joint operation of multiple memristors can be interpreted as the collective function of ion channels in biological synapses (Indiveri et al., 2013). Concretely, we propose the *compound memristive synapse model* which employs M bistable memristors operating in parallel to form a single synaptic weight between two neurons. To implement synaptic plasticity, we employ standard STDP pulsing schemes (Querlioz et al., 2011; Serrano-Gotarredona et al., 2013) and exploit the stochastic nature of memristive switching (Jo et al., 2009b; Gaba et al., 2013; Suri et al., 2013; Yu et al., 2013). For the analysis of the resulting plasticity dynamics, we perceive individual memristors as binary stochastic switches. This abstract description was previously utilized to capture the most salient features of experimentally observed memristive switching (Suri et al., 2013) and appears compatible with pivotal aspects of the experimental literature (Jo et al., 2009b; Gaba et al., 2013).

We show analytically and through computer simulations that the change of the synaptic efficacy for a given pairing of pre- and postsynaptic spikes follows an STDP-like plasticity rule such that the expected weight change depends on the momentary synaptic weight in a stabilizing manner. The resulting *compound-synapse STDP* enables a synapse to attain many memristive states depending on the history of pre- and postsynaptic activity. A stabilizing weight dependence of synaptic plasticity exists in biological synapses (Bi and Poo, 1998) and has been shown to facilitate learning and adaptation in neural systems (Van Rossum et al., 2000; Morrison et al., 2007). In particular, it has been shown in Nessler et al. (2013) that in stochastic winner-take-all (WTA) architectures, STDP with stabilizing weight dependence implements an online Expectation-Maximization algorithm. When exposed to input examples, neurons in the WTA network learn to represent the hidden causes of the observed input in a well-defined generative model. This adaptation proceeds in a purely unsupervised manner. We adopt a similar strategy here and apply the compound memristive synapse model in a network of stochastically spiking neurons arranged in a WTA architecture. We show analytically that compound-synapse STDP optimizes the synaptic efficacies such that the WTA network neurons in the hidden layer represent the most salient features of the input distribution in a Mixture-of-Gaussians generative model. After training, the network performs Bayesian inference over the hidden causes for the given input pattern. We show in computer simulations that such networks are able to learn to represent high-dimensional distributions over images of handwritten digits. After unsupervised training, the network transforms noisy input spike-patterns into a sparse and reliable spike code that supports classification of images. It turns out that even small compound synapses, consisting of only four bistable constituents per synapse, are sufficient for reliable image classification in our simulations. We furthermore show that the proposed model is able to represent the input distribution with high fidelity even in the presence of substantial device variations and under severe noise conditions. These findings render the compound synapse model a promising design principle for novel high-density, low-power mixed-signal CMOS architectures.

2. RESULTS

2.1. STOCHASTIC MEMRISTORS AS PLASTIC SYNAPSES

Memristors have gained increasing attention in neuromorphic engineering as possible substrates for plastic synapses (Jo et al., 2010) due to the possibility to change their electrical conductance without the requirement of extensive supporting circuitry. Recently, Zamarreño-Ramos et al. (2011) and Querlioz et al. (2011) have proposed a pulsing scheme to realize spike-timing dependent plasticity (STDP) with memristive synapses in response to pre- and postsynaptic activity as sketched in **Figure 1A**: Pre-synaptic spikes trigger a rectangular voltage pulse of duration τ (shown in green) that is sent to the memristor's presynaptic terminal. Similarly, postsynaptic neurons send back a copy of their spikes to the postsynaptic terminal as a brief voltage pulse (shown in blue). The combined effect of these pulses was shown to trigger STDP-type plasticity in the memristor as illustrated in **Figure 1B**, where we adopted the convention to measure the voltage drop at the memristor as "presynaptic minus postsynaptic potential." If only the postsynaptic neuron spikes (**Figure 1B**, left), the voltage pulse exceeds the lower threshold of the memristor, leading to long-term depression (LTD). Conversely, if a postsynaptic spike follows a presynaptic spike within duration τ (**Figure 1B**, right), the combined voltage trace exceeds the memristor's upper threshold, thus resulting in long-term potentiation (LTP). Pre-synaptic pulses alone do not trigger any plasticity. Overall, the memristor's conductance obeys an STDP-type plasticity rule.

The direct implementation of the above plasticity rule in mixed-signal neuromorphic architectures, however, faces practical challenges due to the continuous spectrum of the conductance values it relies on. Memristors that support a (quasi-) continuous spectrum of memristive states often suffer from instabilities to maintain their conductance value ("volatility") and typically show unreliable changes under repetitive application of identical pulses.

Here we explore an alternative approach that employs multiple bistable memristors, that support only two distinct conductance states per memristor, to jointly form one synapse. Such devices were reported to exhibit a high degree of uniformity (Fang et al., 2011). Concretely, we propose a synapse model which employs

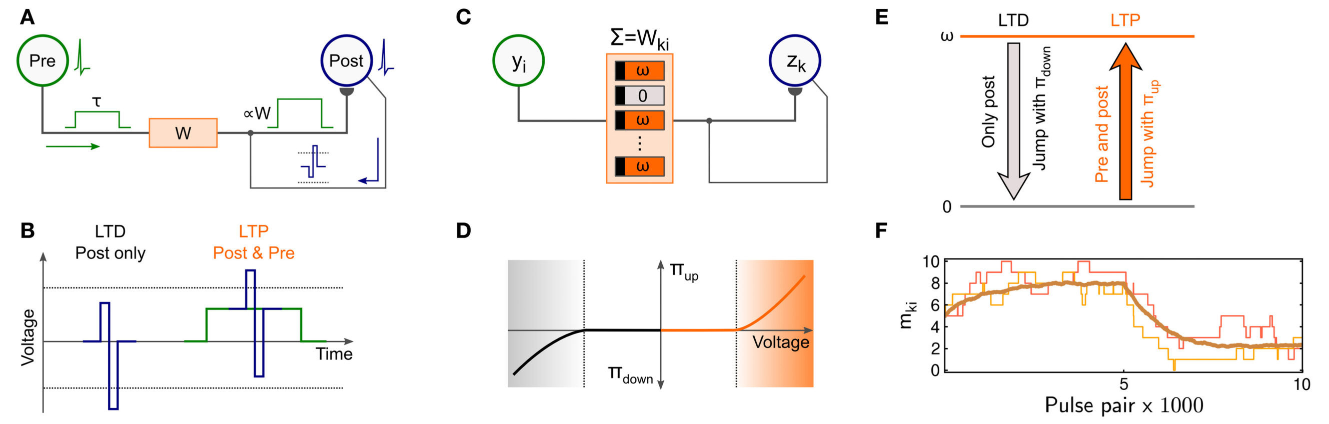


FIGURE 1 | Compound memristive synapse model with stochastic memristors. (A) STDP pulsing scheme. Input spikes elicit a rectangular voltage trace (green, left) that is sent to the presynaptic terminal of the memristor synapse. Post-synaptic spikes elicit a brief voltage pulse (blue) which is sent back to the synapse. **(B)** Solitary postsynaptic spikes trigger LTD since the voltage exceeds the lower threshold of the memristor. Simultaneous pre- and postsynaptic spikes trigger LTP since the voltage exceeds the upper threshold. **(C)** Compound memristive synapse model. A synapse is composed of M bistable memristors operating in parallel. Each memristor can either be active (weight ω) or inactive (weight 0). The total synaptic weight W_{ki} between input neuron y_i and network neuron z_k is the sum of the individual memristor weights. **(D)** Bistable memristors switch stochastically between the active and inactive state depending on the applied voltage difference across its terminals. Switching to the active state (inactive state) occurs with probability π_{up} (π_{down}) if a certain threshold voltage (dotted line) is exceeded. **(E)** Summary of stochastic transitions for compound-synapse STDP. **(F)** In an STDP pairing experiment, the stabilizing weight dependence of compound synapse plasticity governs convergence to a dynamic equilibrium. 10,000 plasticity pulses were applied to a synapse with $M = 10$ constituents. During the first half, 80% (20%) of the events were of LTP (LTD) type. During the second half, the probability for LTP (LTD) events was inverted to 20% (80%). Thin lines: number of active memristors $m_{ki}(t)$ for two example simulation runs. Thick line: Average $\langle m_{ki}(t) \rangle$ over 100 runs. The average weight converges to a dynamic equilibrium.

M bistable memristors operating in parallel to form a single synaptic weight W_{ki} between the i-th input neuron y_i and the k-th network neuron z_k. The model which we refer to as *compound memristive synapse* is sketched in **Figure 1C**. Each memristor is assumed to provide two stable states: a high-conductive (active) state and a low-conductive (inactive) state. Since the dynamic range of memristors typically covers several orders of magnitude, the weight contribution of inactive memristors is almost negligible. In line with this notion, each inactive memristor contributes weight 0 in the synapse model and each active memristor contributes weight ω. Since parallel conductances sum up, the total weight of the compound memristive synapse reads

$$W_{ki} = \omega \cdot m_{ki} \qquad (1)$$

with $m_{ki} \in \{0, 1, \ldots, M\}$ denoting the number of active memristors. As a consequence, a compound memristive synapse supports $M + 1$ discrete weight levels, ranging from 0 to the maximum weight $W_{\max} = \omega \cdot M$.

Plasticity in this synapse model naturally emerges from transitions between the active and inactive state of the bistable constituents. However, deterministic transitions, which are, for instance, desirable in memristor-based memory cells, impair the performance in a neuromorphic online learning setup with compound memristive synapses: If all constituents of a synapse, that experience the same pre- and postsynaptic spikes, change their state simultaneously, the compound weight toggles between $W_{ki} = 0$ and $W_{ki} = W_{\max}$ depending on the latest pulse pair, not showing any gradual trace of memory formation as required for STDP.

A possible remedy to this issue can be found in memristors that exhibit stochastic rather than deterministic switching between their stable states. Yu et al. (2013), for instance, reported stochastic transitions in HfO_x/TiO_x memristors (from the class of anion-based memristors) and explored in computer simulations how stochastic bistable memristors could be used in a neuromorphic learning architecture. Similar studies explored the usability of stochastically switching bistable cation-based memristive materials (Jo et al., 2009b; Gaba et al., 2013; Suri et al., 2013). In these nanoscale devices, changes in the memristance were shown to be dominated by the formation of a single conductive filament (Jo et al., 2009b). Stochastic switching was demonstrated for both directions (active ↔ inactive) (Suri et al., 2013) with switching probabilities being adjustable via the duration (Gaba et al., 2013; Suri et al., 2013) and amplitude (Jo et al., 2009b; Gaba et al., 2013) of the voltage applied across the terminals. These observations have led to the conclusion that "switching can be fully stochastic" with switching probabilities being almost unaffected by the rate at which consecutive plasticity pulses are delivered (Gaba et al., 2013). Furthermore, bistable memristors can be extremely durable, not showing any notable degradation over hundreds of thousands of programming cycles (Jo et al., 2009b). Owing to these pivotal properties, Suri et al. (2013) proposed an STDP-type plasticity rule that perceives bistable memristors as simple stochastic switches.

Here we generalize this idea to compound memristive synapses and investigate the computational function of the arising plasticity rules in spiking networks from a machine learning perspective. **Figure 1D** illustrates a simple model of stochastic switching of individual memristors that employs the STDP-pulsing scheme from Zamarreño-Ramos et al. (2011) and Querlioz et al. (2011) discussed above: If the voltage between the pre- and postsynaptic terminal of a device exceeds a certain threshold (dotted lines) for the duration of the back-propagating spike signal, stochastic switching may occur. Inactive memristors jump to the active state with probability π_{up} provided a sufficiently strong positive voltage, resulting in stochastic LTP. Similarly, active memristors turn inactive with probability π_{down} given a sufficiently strong negative voltage, leading to stochastic LTD. No switching occurs if only a small (or zero) voltage is applied, or if the memristor is already in the respective target state. Since the applied pre- and postsynaptic pulse amplitudes are free parameters in the model, the jumping probabilities π_{up} and π_{down} can be controlled, to a certain extent, by the experimenter. This simple model captures the most salient aspects of stochastic switching in bistable memristive devices as discussed above, cp. also Suri et al. (2013) and the Discussion section. In order to distinguish the abstract memristor model from physical memristive devices, we will in the following refer to the model memristors as "stochastic switches," "constituents," or simply as "switches" for the sake of brevity. Furthermore, we refer to the resulting stochastic plastic behavior of compound memristive synapses in response to pre-post spike pairs, summarized in **Figure 1E**, as *compound-synapse STDP*.

From the transition probabilities of individual stochastic switches we can calculate the expected temporal weight change $\langle \frac{d}{dt} W_{ki} \rangle$ of the compound memristive synapse as a function of pre- and postsynaptic activity. Formally, we denote the presence of a rectangular input pulse (green in **Figure 1A**) of the i-th input by $y_i(t) = 1$ (and the absence by $y_i(t) = 0$). The brief pulses that are sent back from a postsynaptic neuron z_k to the synapse (blue in **Figure 1A**) are formally treated as point events at the spike times of the postsynaptic neuron. We denote the spike time of the f^{th} spike of neuron z_k by t_k^f. The spike train $s_k(t)$ of a neuron z_k is formally defined as the sum of Dirac delta pulses $\delta(\,\cdot\,)$ at the spike times: $s_k(t) = \sum_f \delta(t - t_k^f)$. When a synaptic efficacy W_{ki} is subject to a stochastic LTP update, there are $(M - m_{ki})$ constituents in the compound memristive synapse that are currently inactive and could undergo an LTP transition. Each constituent independently switches to its active state with probability π_{up}, thereby contributing ω to the compound weight W_{ki}. Hence, the expected weight change for the LTP condition reads $(M - m_{ki})\,\omega\,\pi_{\text{up}}$. A similar argumentation applies to the LTD case. In summary, considering that plasticity always requires a postsynaptic spike, and that LTP is induced in the presence of a presynaptic pulse ($y_i(t) = 1$), while LTD is induced in the absence of a presynaptic pulse ($y_i(t) = 0$), the expected weight change of the compound memristive synapse reads:

$$\begin{aligned}
\langle \frac{d}{dt} W_{ki} \rangle &= s_k(t) \cdot \Bigg[\underbrace{(M - m_{ki})\,\omega\,\pi_{\text{up}}\,y_i(t)}_{\text{LTP}} \\
&\qquad - \underbrace{m_{ki}\,\omega\,\pi_{\text{down}}\,(1 - y_i(t))}_{\text{LTD}} \Bigg] \qquad (2) \\
&= s_k(t) \cdot \big[M\omega\pi_{\text{up}}\,y_i(t) - m_{ki}\omega\pi_{\text{up}}\,y_i(t) \\
&\qquad - m_{ki}\omega\pi_{\text{down}} + m_{ki}\omega\pi_{\text{down}}\,y_i(t) \big] \\
&= s_k(t) \cdot \big[W_{\max}\,\pi_{\text{up}}\,y_i(t) - W_{ki}\pi_{\text{down}} \big] \\
&\qquad + s_k(t)\,W_{ki}\,y_i(t) \cdot (\pi_{\text{down}} - \pi_{\text{up}}) \quad .
\end{aligned}$$

In order to obtain a simple closed form solution of the weight changes, we set $\pi_{\text{up}} = \pi_{\text{down}}$, i.e., the probability of potentiation of a single switch under LTP equals its probability of depression under LTD. This choice will facilitate the theoretical analysis of learning in a spiking network, later on. In a hardware implementation, the switching probabilities could be adjusted via the pre- and postsynaptic amplitudes of the STDP pulses. We obtain the following closed form solution for the expected weight change:

$$\langle \frac{d}{dt} W_{ki} \rangle = \pi_{\text{up}}\,W_{\max} \cdot s_k(t) \cdot \big[y_i(t) - W_{ki}/W_{\max} \big] \quad . \qquad (3)$$

Notably, the plasticity rule (3) differs from standard additive STDP rules in that it includes the weight dependent term $W_{ki}/W_{\max}$. This weight dependence has its origin in the varying number of (in-)active stochastic switches m_{ki} that could actually undergo plastic changes and is in line with a prominent finding from neurobiology (Bi and Poo, 1998): Relative changes $\Delta W_{ki}/W_{ki}$ become weaker for strong weights under LTP, while under LTD the relative change is weight-independent. Studies in computational neuroscience, see e.g., Van Rossum et al. (2000), found that this type of weight dependence facilitates the formation of stable connections in spiking networks.

In **Figure 1F**, we illustrate the stochastic convergence of a compound synaptic weight to a stable, dynamic equilibrium in a simple STDP pairing experiment. The synapse consists of $M = 10$ bistable switches with switching probabilities set to $\pi_{\text{up}} = \pi_{\text{down}} = 0.001$. These synapse parameters will also be used in network level simulations, later on. In a small computer simulation, 5 of the 10 constituents are initially set active ($m_{ki}(t = 0) = 5$). Then, 5000 postsynaptic spikes are sent to the postsynaptic terminal of the synapse for triggering stochastic switching in the bistable constituents. 80% of these events are randomly paired with a presynaptic pulse, i.e., 80% of the events are of LTP type and 20% are of LTD type. After 5000 plasticity pulses, the statistics are inverted to 20% LTP and 80% LTD events for another 5000 plasticity pulses. The thin lines in **Figure 1F** show the evolution of $m_{ki}(t)$ in this STDP pairing experiment for two independent simulation runs. In the first half, the synaptic weight tends to settle around higher weight values, while in the second half, it stochastically declines toward lower weight values. The gradual convergence of the average weight, as suggested by Equation (3), becomes apparent by taking the mean over 100 simulation runs (thick line). The stabilizing weight dependence of the STDP rule leads to convergence to a dynamic equilibrium such that the mean value shows slow, continuous changes as expected from the theory. Individual synapses fluctuate stochastically around this mean.

Equation (3) is reminiscent of a theoretically derived synaptic plasticity rule for statistical model optimization in spiking neural networks proposed by Nessler et al. (2013). Building upon the theoretical approach developed by Nessler et al. (2013), we will next turn to the question how to conceive stochastic learning with compound memristive synapses from a Bayesian perspective as unsupervised model optimization via a powerful optimization method that is known as Expectation-Maximization in the machine learning literature.

2.2. COMPOUND MEMRISTIVE SYNAPSES IN WINNER-TAKE-ALL NETWORKS

The winner-take-all (WTA) network structure is a ubiquitous circuit motif in neocortex (Douglas and Martin, 2004; Lansner, 2009) and is often utilized in neuromorphic engineering (Mead and Ismail, 1989; Indiveri, 2000). Recently, WTA networks attracted increasing attention in theoretical studies on statistical learning (Habenschuss et al., 2012; Keck et al., 2012; Habenschuss et al., 2013; Nessler et al., 2013; Kappel et al., 2014) because their comparatively simple network dynamics facilitate a comprehensive mathematical treatment. In this section, we introduce the WTA architecture that we will use to study the learning capabilities of spiking neural networks with compound memristive synapses subject to STDP.

The WTA network architecture is sketched in **Figure 2A**. The network consists of N spiking input neurons $y_1, \ldots, y_N$ and K spiking network neurons $z_1, \ldots, z_K$ with all-to-all connectivity in the forward synapses. Lateral inhibition introduces competition among the network neurons. Network neurons are stochastic spike response neurons (Gerstner and Kistler, 2002) with the membrane potential u_k of network neuron z_k being given by

$$u_k(t) = b_k + \sum_{i=1}^{N} W_{ki} \cdot y_i(t) \quad . \tag{4}$$

The membrane potential $u_k(t)$ integrates the inputs $y_i(t)$, i.e., the rectangular voltage pulses following each input spike, linearly through the synaptic weights W_{ki}. The parameter b_k denotes the intrinsic excitability of the neuron and controls its general disposition to fire. In Methods we outline how the linear membrane potential (4) can be realized with leaky integrators, a common neuron model in neuromorphic designs. For the spike response of the network neurons z_k, a stochastic firing mechanism is employed. In the WTA network, neurons z_k spike in a Poissonian manner with instantaneous firing rate $\rho_k(t)$ that depends on the membrane potential $u_k(t)$ and on lateral inhibition $u_{\text{inh}}(t)$:

$$\rho_k(t) = r_{\text{net}} \cdot e^{u_k(t) - u_{\text{inh}}(t)} \quad , \tag{5}$$

with a constant $r_{\text{net}} > 0$ that scales the overall firing rate of the network. In other words, the neuron spikes with probability $\rho_k(t) \cdot \delta t$ in a small time window $\delta t \to 0$. The inhibitory contribution $u_{\text{inh}}(t) := \log \sum_{j=1}^{K} \exp(u_j(t))$ summarizes the effect of lateral inhibition in the network and introduces WTA-competition between the network neurons to fire in response to a given stimulus $y_1(t), \ldots, y_N(t)$. Notably, the exponential relationship (5) between an idealized membrane potential and neuronal firing is consistent with biological findings about the response properties of neocortical pyramidal neurons (Jolivet et al., 2006).

The feed-forward synapses from inputs y_i to network neurons z_k are implemented as compound memristive synapses and their synaptic weights W_{ki} are adapted through stochastic STDP as described above. The intrinsic excitabilities b_k are adapted according to a homeostatic plasticity rule (Habenschuss et al., 2012) that ensures that all network neurons take part in the network response and thus facilitates the emergence of a rich neural representation that covers the entire input space. Besides its observed stabilizing effect (Querlioz et al., 2011), homeostatic intrinsic plasticity plays a distinct computational role when combined with synaptic learning: Network neurons that maintain many strong synapses W_{ki} gain an "unrightful advantage" during WTA competition over neurons that are specialized on low-activity input patterns (and therefore maintain weaker weights). A detailed analysis of the learning dynamics in the network shows that, in order to compensate for this advantage, the former must be burdened with a lower excitability b_k than the latter. A formal definition of the homeostatic plasticity mechanism and a discussion of its computational role from a theory

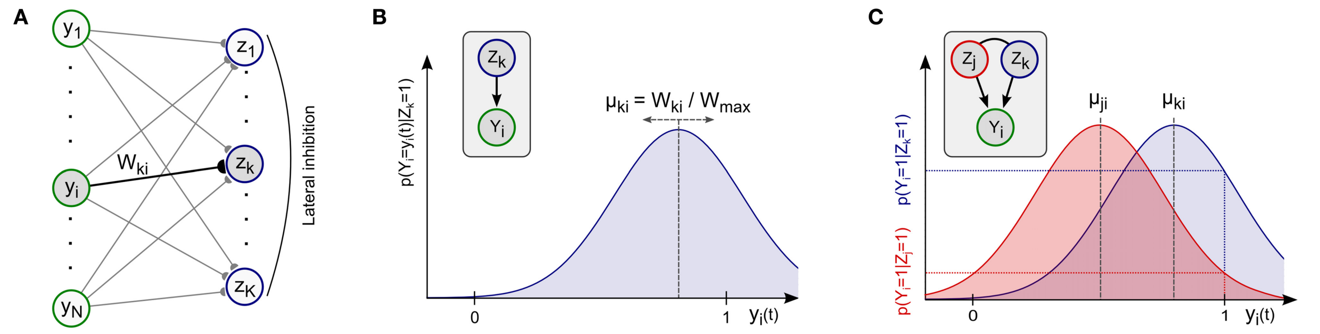


FIGURE 2 | Spiking network for probabilistic inference and online learning. (A) Winner-take-all network architecture with lateral inhibition and synaptic weights W_{ki}. **(B)** Network neurons z_k implicitly maintain a Gaussian likelihood function for each input y_i in their afferent synaptic weights W_{ki}. The mean μ_{ki} of the distribution is encoded by the fraction $W_{ki}/W_{\max} = m_{ki}/M$ of active switches in the compound memristive synapse, i.e., stronger synaptic weights W_{ki} correspond to higher mean values μ_{ki}. Inset: Local implicit graphical model. **(C)** Illustration of Bayesian inference for two competing network neurons z_k, z_j and one active input $y_i(t) = 1$. Different means μ_{ki}, μ_{ji} encoded in the weights give rise to different values in the likelihood function and shape the posterior distribution according to Bayes rule.

perspective are provided in Methods, see also Habenschuss et al. (2012).

2.3. MEMRISTIVE SYNAPSES SUPPORT INFERENCE AND ONLINE LEARNING

In this section, we thoroughly analyze the learning effects of STDP in compound memristive synapses in the stochastic WTA network model. For the mathematical analysis, we describe the inputs $y_i(t)$ and the stochastic neuron responses $s_k(t)$ with the help of probability theory. To this end, we perceive the spiking activity of the input neurons y_i and network neurons z_k as samples of random variables (RVs) Y_i and Z_k respectively. Consistent with the assumption that spikes from input neurons produce voltage pulses of duration τ in the circuit implementation (see **Figure 1A**), we set $Y_i = y_i(t)$, i.e., $Y_i = 1$ if input neuron y_i spiked within $[t - \tau, t]$ and $Y_i = 0$ otherwise. The assignment of output spikes $s_k(t)$ to RVs Z_k is different: The random variable Z_k labels the winner of the WTA network at spike times of network neurons. Hence, the value of Z_k is only defined at the moments when one of the K network neurons spikes, i.e., when the spike train $s_j(t) \neq 0$ for some neuron z_j. In this case, Z_k encodes which neuron spiked, and we set $Z_k = 1$ if $k = j$ and $Z_k = 0$ if $k \neq j$.

Using this interpretation of neural activity as realizations of RVs, the network's stochastic response $s_k(t)$ to an input configuration $\boldsymbol{y}(t) = (y_1(t), \ldots, y_N(t))$ gives rise to a conditional probability distribution $p_{\text{net}}(\boldsymbol{Z} \,|\, \boldsymbol{Y})$ over the network RVs $\boldsymbol{Z} := (Z_1, \ldots, Z_K)$ conditioned on the input RVs $\boldsymbol{Y} := (Y_1, \ldots, Y_N)$. In line with the definition of the RVs Z_k, the distribution $p_{\text{net}}(\boldsymbol{Z} \,|\, \boldsymbol{Y})$ describes the network response only when one of the network neurons z_k fires. The response distribution $p_{\text{net}}(\boldsymbol{Z} \,|\, \boldsymbol{Y} = \boldsymbol{y}(t))$ for any fixed input configuration $\boldsymbol{y}(t)$ can directly be calculated from Equations (4) and (5) (note that the probability $p_{\text{net}}(Z_k = 1 \,|\, \boldsymbol{Y} = \boldsymbol{y}(t))$ for an individual RV Z_k to be active is proportional to the firing rate $\rho_k(t)$ of neuron z_k):

$$p_{\text{net}}(Z_k = 1 \,|\, \boldsymbol{Y} = \boldsymbol{y}(t)) = \rho_k(t)/r_{\text{net}} = e^{u_k(t) - u_{\text{inh}}(t)} = \frac{e^{b_k + \sum_{i=1}^{N} W_{ki} \cdot y_i(t)}}{\sum_{j=1}^{K} e^{b_j + \sum_{i=1}^{N} W_{ji} \cdot y_i(t)}} \quad . \tag{6}$$

Equation (6) fully characterizes the network response distribution $p_{\text{net}}(\boldsymbol{Z} \,|\, \boldsymbol{Y})$ for any given input $\boldsymbol{Y} = \boldsymbol{y}(t)$. We next turn to the question how the response distribution $p_{\text{net}}(\boldsymbol{Z} \,|\, \boldsymbol{Y})$ can be understood as the result of a meaningful probabilistic computation. Specifically, we will show that the spike response of the WTA network approximates the Bayesian posterior distribution during inference in a well-defined probabilistic model. This probabilistic model is implicitly encoded in the synaptic weights W_{ki}, and synaptic plasticity can thus be perceived as an ongoing refinement of the involved probability distributions. Indeed, the STDP rule (3) of the compound memristive synapses turns out to be optimal in the sense that it instantiates *generalized Expectation-Maximization* in the WTA network, a powerful algorithm for unsupervised learning from machine learning theory. Our findings build upon theoretical work on synaptic learning in spiking neural networks from Nessler et al. (2009, 2013) and Habenschuss et al. (2012, 2013).

The key idea for identifying the response distribution $p_{\text{net}}(\boldsymbol{Z} \,|\, \boldsymbol{Y})$ as the result of a Bayesian computation, is to hypothetically reverse the network computation and view the spike response of a network neuron z_k as the *hidden cause* behind the observed input $\boldsymbol{y}(t)$. In this view, the network is treated as a *generative model* that implicitly defines a prior distribution $p(\boldsymbol{Z})$ over hidden causes Z_k and a set of likelihood distributions $p(\boldsymbol{Y}|Z_k = 1)$, one for each hidden cause Z_k. The shape of the distributions is defined by the parameters of the network, e.g., the synaptic weights W_{ki}. An important property of these implicitly encoded distributions–that also motivates the term "generative model"–is that they give rise to a (hypothetical) distribution over the inputs, $p(\boldsymbol{Y} = \boldsymbol{y}(t)) = \sum_{k=1}^{K} p(\boldsymbol{Y} = \boldsymbol{y}(t)|Z_k = 1) \cdot p(Z_k = 1)$. This equation also explains why a RV Z_k is called a hidden cause: If we observe an input vector $\boldsymbol{y}(t)$, that has high probability only in one of the likelihood distributions $p(\boldsymbol{Y} = \boldsymbol{y}(t)|Z_k = 1)$, then we can consider the RV Z_k as a likely (but unobservable) cause for the observation according to the generative model. A common objective in machine learning theory, known as *Maximum-Likelihood learning*, is to find parameters that bring the implicit distribution $p(\boldsymbol{Y})$ of the generative model as close as possible to the distribution of the actually observed input. Then the hidden causes of the generative model are expected to represent important features of the observed input (e.g., some typical input clusters). Leaving the hypothetical generative perspective again, in the network's real operation an input $\boldsymbol{y}(t)$ is presented to the network and the hidden causes Z_k need to be inferred (e.g., the cluster the input $\boldsymbol{y}(t)$ belongs to). The mathematically correct result of this inference is given by Bayes rule

$$p(Z_k = 1 \,|\, \boldsymbol{Y} = \boldsymbol{y}(t)) \propto p(Z_k = 1) \cdot p(\boldsymbol{Y} = \boldsymbol{y}(t) \,|\, Z_k = 1) \tag{7}$$

which combines the likelihood $p(\boldsymbol{Y} = \boldsymbol{y}(t) \,|\, Z_k = 1)$ with the prior $p(Z_k = 1)$.

Nessler et al. (2013) showed that in a WTA network architecture, that evolves according to Equations (4) and (5), the synaptic weights W_{ki} can be understood as an implicit neural encoding of likelihood distributions $p(\boldsymbol{Y} \,|\, Z_k = 1)$, and that the network response $p_{\text{net}}(\boldsymbol{Z} \,|\, \boldsymbol{Y})$ approximates the posterior distribution according to Equation (7). Hence, WTA networks can be regarded as implicit generative models. Furthermore–and even more importantly from a theoretical perspective–Nessler et al. (2013) showed that the implicit likelihood model, that is encoded in the weights W_{ki}, can be optimized in an unsupervised manner by a weight-dependent STDP rule. Indeed, there exists a tight link between the type of weight dependence in the STDP rule and the type of implicit likelihood model it optimizes. The exponential weight dependence in Nessler's rule, however, differs from the linear weight dependence we identified for the plasticity rule in Equation (3). This raises the question what type of implicit likelihood model is encoded and optimized by the compound memristive synapses.

An intuition about an appropriate probabilistic interpretation of compound-synapse STDP can be obtained from the equilibrium points of the plasticity rule (3). We first observe, that plasticity is always triggered by a postsynaptic spike, i.e., $Z_k = 1$ for some neuron z_k. In the spirit of spike-triggered averaging (Simoncelli et al., 2004), we can then study the conditional distribution $p(Y_i = y_i(t) \mid Z_k = 1)$ of an input y_i at the moment of the network response since the coincidence of pre- and postsynaptic spiking activity is the driving force behind any weight change in the STDP rule. By assuming that plasticity has converged to a dynamic equilibrium, the average contributions of LTP and LTD cancel each other, i.e., $\langle \frac{d}{dt} W_{ki} \rangle_{p(Y_i \mid Z_k = 1)} = 0$. From this condition, we obtain the following relation between the synaptic weight W_{ki} and the conditional input distribution $p(Y_i \mid Z_k = 1)$:

$$\begin{aligned} 0 &\overset{!}{=} \langle \frac{d}{dt} W_{ki} \rangle_{p(Y_i \mid Z_k = 1)} \\ &= \langle \pi_{\text{up}} \, W_{\max} \cdot [Y_i - W_{ki}/W_{\max}] \rangle_{p(Y_i \mid Z_k = 1)} \\ \Rightarrow \quad W_{ki} &= W_{\max} \cdot \langle Y_i \rangle_{p(Y_i \mid Z_k = 1)} \quad . \end{aligned} \tag{8}$$

According to this analysis, the synaptic weight W_{ki} represents the expected value of input neuron y_i at the moment of a postsynaptic spike in network neuron z_k in a linear manner. In the compound memristive synapse, this expectation is encoded in the $M + 1$ possible weight states $W_{ki} = 0, \omega, \ldots, W_{\max}$ of the synapse. The linear encoding (8) is compatible with the convergence points which we observed previously in the small STDP pairing experiment in **Figure 1F**.

The above analysis only serves as an intuition and is no substitute for a thorough mathematical treatment of the learning process. A rigorous formal derivation that, for instance, also takes into account the dynamically changing response properties of the network neurons due to recurrent interactions and plastic changes in the weights, is provided in Methods. It reveals that the above intuition holds. More precisely, the likelihood distributions $p(\boldsymbol{Y} \mid Z_k = 1)$ that are optimized in a WTA circuit with compound-synapse STDP are given by the product of the likelihoods of individual inputs

$$p(\boldsymbol{Y} = \boldsymbol{y}(t) \mid Z_k = 1) = \prod_{i=1}^{N} p(Y_i = y_i(t) \mid Z_k = 1) \quad , \tag{9}$$

and the likelihood for each individual input channel y_i is given by a Gaussian distribution

$$p(Y_i = y_i(t) \mid Z_k = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(y_i(t) - \mu_{ki})^2}{2\sigma^2}} \quad . \tag{10}$$

The mean values μ_{ki} and the standard deviation σ of the likelihood distributions (10) are identified as

$$\mu_{ki} = W_{ki}/W_{\max} = m_{ki}/M \qquad \text{and} \qquad \sigma = 1/\sqrt{(W_{\max})} \quad . \tag{11}$$

Hence, the mean μ_{ki} of the distribution for input channel y_i is given by the fraction of active constituents in the compound memristive synapse. The width $\sigma = 1/\sqrt{(W_{\max})}$ of the distribution is determined by the maximum weight $W_{\max} = M \cdot \omega$ and could, for instance, be controlled by the weight contribution ω of an individual stochastic switch. The resulting probabilistic model of the WTA network is a *Mixture-of-Gaussians* generative model (see Methods for a formal definition).

In order to illustrate the computational properties of this generative model, the likelihood distribution $p(Y_i \mid Z_k = 1)$ for a single input y_i and a single active hidden cause $Z_k = 1$ is sketched in **Figure 2B**. An active hidden cause $Z_k = 1$ assigns probabilities to all possible instantiations $y_i(t)$ of Y_i. In principle, the Gaussian likelihood distribution supports arbitrary real-valued input instantiations $y_i(t) \in \mathbb{R}$. We will come back to this observation in the Discussion section where we address possible extensions of the WTA network to support more complex input types. In this article, we consider only binary inputs that take on the value 0 (input pulse absent) or 1 (input pulse present), see the presynaptic pulses in **Figure 1A**. The corresponding likelihood values $p(Y_i = 0 \mid Z_k = 1)$ and $p(Y_i = 1 \mid Z_k = 1)$ are determined by the mean μ_{ki} and the variance σ of the likelihood distribution, see Equations (10) and (11). The task of the network when presented with an input $\boldsymbol{y}(t)$ is to infer the posterior distribution over hidden causes $p(Z_k = 1 \mid \boldsymbol{Y} = \boldsymbol{y}(t))$ and produce spikes according to this distribution. The optimal solution is given by Bayes rule (7) with the likelihood $p(\boldsymbol{Y} = \boldsymbol{y}(t) \mid Z_k = 1)$ given by Equations (9) and (10), and an (input independent) a priori probability $p(Z_k = 1)$. As we prove in Methods, the response distribution $p_{\text{net}}(\boldsymbol{Z} \mid \boldsymbol{Y} = \boldsymbol{y}(t))$ of the spiking WTA network implements a close and well-defined variational approximation of this posterior distribution. A minimal example of such Bayesian inference is sketched in **Figure 2C**, where we consider a small WTA network with only one input y_i and two network neurons z_k and z_j. For a given input instantiation $y_i(t)$ the values $p(Y_i = y_i(t) \mid Z_k = 1)$ and $p(Y_i = y_i(t) \mid Z_j = 1)$ measure the likelihoods of the two competing hypotheses that neuron z_k or neuron z_j is the hidden cause of the observed input $y_i(t)$. These likelihood values shape the Bayesian posterior distribution (7) by contributing one factor to the product in Equation (9).

As a consequence, a network neuron z_k is particularly responsive to those input configurations $\boldsymbol{y}(t)$ that are associated with high likelihood values $p(\boldsymbol{Y} = \boldsymbol{y}(t) \mid Z_k = 1)$. The likelihood distributions are determined by the means μ_{ki}, i.e., by the number m_{ki} of active switches in the synapses that change according to the compound-synapse STDP rule. Through this mechanism, synaptic plasticity governs the emergence of prototypic patterns that the network neurons are most responsive to, and thereby turns each neuron z_k into a probabilistic expert for certain input configurations $\boldsymbol{y}(t)$. The aim of learning in the WTA network is to distribute the probabilistic experts z_k such that the likelihood of the presented input is (on average) as high as possible. This objective is an equivalent formulation of Maximum-Likelihood learning, and the *log-likelihood function*, which measures the average (logarithm of the) input likelihood, is a widely-used measure to determine how well a learning system is adapted to the presented input. Formally, the learning process can be described within the framework of generalized Expectation-Maximization (Dempster et al., 1977; Habenschuss et al., 2012; Nessler et al., 2013). In Methods we show that the generalized Expectation-Maximization algorithm is implemented in the WTA network via the interplay of the compound-synapse STDP rule (3), that adapts the synaptic weights W_{ki}, and the homeostatic intrinsic plasticity rule, that regulates the intrinsic excitability b_k of the neurons such that each neuron maintains a long-term average firing rate. The interplay of synaptic and intrinsic plasticity achieves the aim of Maximum-Likelihood learning in the WTA network in the following sense:

The expected synaptic weight changes $\langle \frac{d}{dt} W_{ki} \rangle$ *of the compound memristive synapses on average increase a lower bound of the log-likelihood function in a Mixture-of-Gaussians generative model during online learning until a (local) optimum is reached.*

2.4. DEMONSTRATION OF UNSUPERVISED LEARNING

We tested the learning capabilities of the compound memristive synapse model in a standard machine learning task for hand-written digit recognition. In a computer simulation, we set up a WTA network with $N = 24 \times 24$ input neurons and $K = 10$ network neurons. Each synaptic weight W_{ki} was composed of $M = 10$ stochastic bistable switches, each contributing $\omega = 0.1$ in its active state. The switching probabilities $\pi_{\text{up}} = \pi_{\text{down}} = 0.001$ were set to a quite low value. This corresponds to a long integration time for gradual memory formation in order to assess the general ability of the synapse model in online learning tasks. Before training, the stochastic switches were initialized randomly as shown in **Figure 3A**. The network was then exposed to hand-written digits 0–4 from the MNIST training data set (LeCun et al., 1998). Examples from the data set are shown in **Figure 3B**. Each pixel was encoded by one input neuron y_i. Digits were presented as Poisson spike trains with firing rates depending on pixel intensities. Overall, the network was trained in an unsupervised setup for 5000 s with a new digit being presented every 100 ms. **Figure 3C** shows the weight matrix in an early stage of learning at $t = 200$ s. At this stage, the synapses begin to integrate salient statistical features of the input, such as the generally low activity along the frame. Furthermore, a specialization to certain digit classes becomes apparent for some of the network neurons.

Over the course of learning, the synapses continuously improve the network's implicit generative model of the presented input. This refinement is reflected in the log-likelihood function shown in **Figure 3D** that measures how well the probabilistic model is adapted to the input distribution. The ongoing refinement also becomes apparent by a more intuitive–and practically more relevant–measure, namely the classification performance of the network on an independent test set of handwritten digits. The classification error (blue in **Figure 3D**; see Methods for details) continuously decreases as training progresses. The improved performance on an independent test set furthermore indicates that the network develops a generally well-suited representation of the input and evades the risk of over-fitting.

At the end of training, after $t = 5000$ s, a set of prototypic digits has emerged in the compound memristive synapses as shown in **Figure 3E**. The well-adapted synapse array turns each network neuron into a probabilistic expert for a certain digit class. As a consequence, the network has learned to transform the N-dimensional, noisy spike input into a sparse and reliable spike code, as shown in **Figure 3F**. Typically, exactly one network neuron z_k fires in response to the input. But also the seemingly unclear cases, when two neurons respond simultaneously, carry meaningful information in a Bayesian interpretation: Since network spikes approximate the posterior distribution $p(\mathbf{Z} \,|\, \mathbf{Y} = \mathbf{y}(t))$ through sampling, an ambiguous spike response encodes the level of uncertainty during probabilistic inference.

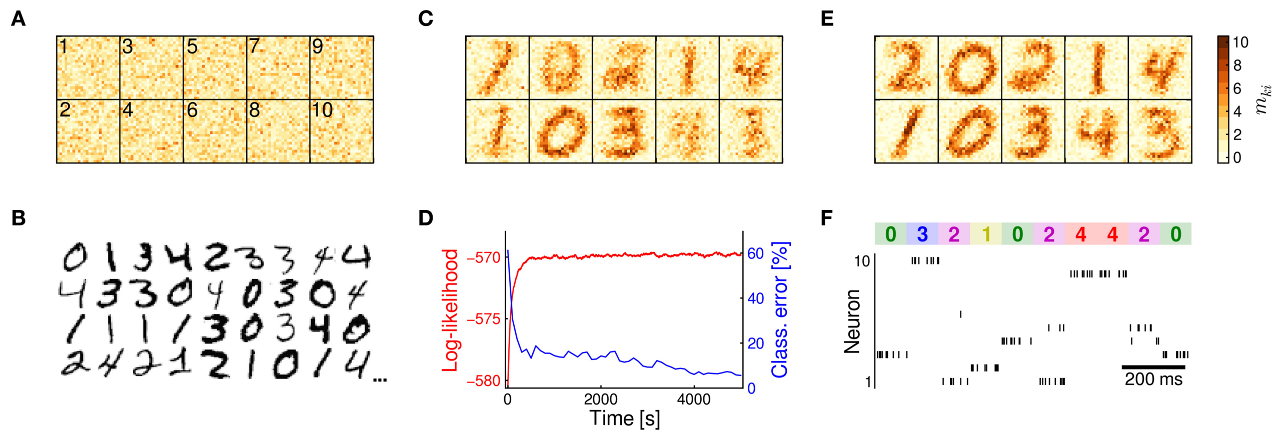


FIGURE 3 | Learning of hand-written digits. (A) Synaptic weights at time $t = 0$ *s* after random initialization. Color intensities show the number $m_{ki} \in [0, 10]$ of active switches for each connection. The indices k of the postsynaptic neurons are indicated in the top-left corners. **(B)** Examples from the MNIST data set. Pixel intensities of the digits were encoded as Poisson spike trains and presented to the network. Digits 0–4 were presented with a new digit being shown every 100 ms. **(C)** Weight matrix at time $t = 200$ *s*. Memristive synapses start to integrate salient features of the input stream. **(D)** Over the course of learning the log-likelihood function (red) increases, indicating that the network continuously refines its implicit statistical model of the presented data. This refinement leads to an improved classification performance (blue) on an independent test set. **(E)** At the end of the learning experiment at $t = 5000$ *s*, the synapses have specialized on different prototypes of the presented digits, rendering each network neuron a probabilistic expert for a certain digit. **(F)** Network response for 1 *s* at the end of learning. The presented digit is shown at the top. The input is transformed into a sparse and reliable spike code. Some digits invoke a spike response of more than one neuron. This ambiguous response encodes uncertainty in the variational posterior distribution.

2.5. INFLUENCE OF SYNAPTIC RESOLUTION

In the previous section, we have demonstated that the compound memristor synapse model is able to learn statistical regularities in the input stream and enables the spiking WTA network to perform probabilistic inference in a well-defined generative model. The demonstration employed compound synapses with $M = 10$ stochastic switches per synapse. Notably, the number of constituents M is a free parameter of the model and determines the weight resolution of the compound synapse. Increasing the synaptic resolution by recruiting more bistable switches per synapse is generally expected to improve the accuracy of the input representation, but comes at the cost of reduced integration density in a neuromorphic design. In the following, we therefore explore the opposite direction, i.e., unsupervised learning with a low weight resolution.

For estimating the influence of the weight resolution on the learning capabilities of the WTA network, we repeated the above computer simulation for different values of M, while holding the maximum weight $W_{\max} = \omega \cdot M$, and thus the variance $\sigma = 1/\sqrt{(W_{\max})}$ of the implicit generative model, fixed. **Figure 4A** shows examples of the digits stored in the synapse array after 5000 s of learning for $M = 1$, 2, 4, and 100 stochastic switches per synapse. Even binary synapses with $M = 1$ successfully identify noisy archetypes of the input digits. This observation is in line with previous studies on learning with binary weights (Fusi, 2002). The accuracy of the representation quickly increases with higher M-values. As an (academic) reference, we also included a simulation with $M = 100$ switches per synapse which support a quasi-continuous state spectrum.

The resulting ability of the WTA network to recognize handwritten digits is shown in **Figure 4B** in terms of the classification error on a test set. Each bar depicts the mean performance of 20 independently trained networks per M-value, errorbars show the standard deviation among networks. Taking the academic example with $M = 100$ as a reference for the performance achievable by the small WTA network, the computer simulations suggest that as few as $M = 4$ constituents per synapse may be sufficient for practical applications. While individual weights W_{ki} only store little information (ca. 2.3 bits in case of $M = 4$) about the expected input in channel y_i, the partial evidence received from each of the $N = 576$ input channels is integrated by the network in a statistically correct manner to form a sharply peaked posterior, most of the time.

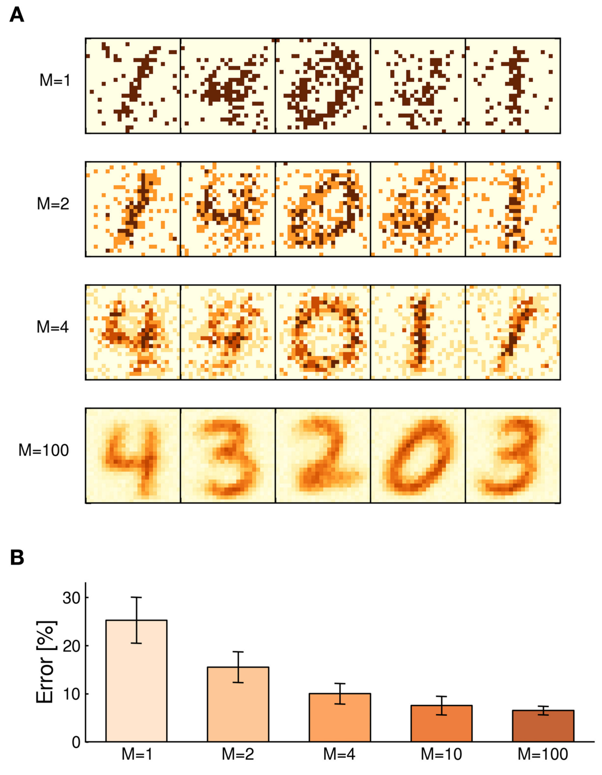


FIGURE 4 | Influence of the synaptic resolution. (A) Examples of learned weight matrices with different numbers of constituents M per synapse. Even binary weights with only one stochastic switch learn to maintain a coarse image of prototypic digits. In the limit of large M, compound synapses support a quasi-continuous weight spectrum. **(B)** Classification error for different weight resolutions after 5000 s of learning, based on 20 independently trained networks for each value of M. Errorbars: *SD* among networks.

2.6. ROBUSTNESS TO DEVICE VARIATIONS

We have demonstrated so far that spiking networks with compound memristive synapses can learn a faithful representation of their input when synapses consist of idealized bistable constituents. Large-scale physical implementations, however, are likely to exhibit substantial device variabilities and imperfections. Plasticity in the compound synapse model depends on two device properties that are likely to be distorted in physical implementations: the conductance of each individual constituent ω and the switching probabilities $\pi_{\text{up}}/\pi_{\text{down}}$. In the following, we address the impact of distortions in these two properties on the WTA network learning capabilities, separately.

We first turned to the conductance value ω of individual switches and investigated the robustness of learning to two fundamentally different types of noise in ω, namely spatial noise and temporal noise:

- *Spatial noise* describes device-to-device variations and addresses peculiarities of individual memristors that remain stable over time.
- *Temporal noise* refers to trial-to-trial variations and covers device instabilities over the course of learning.

Both types of noise can be suspected to induce serious disturbances during learning: In case of spatial noise, although device-to-device variations could average out if many memristors are employed, any remaining deviations give rise to sustained systematic errors that may build up over the course of learning. In case of temporal noise, while trial-to-trial variations could average out over time, any synaptic update rests upon a disturbed instantiation of the weight matrix, i.e., on a noisy (and false) assumption. In computer simulations, we accounted for these types of noise separately by disturbing the active-state weight value ω of the switches as sketched in **Figure 5A**. To model spatial noise, the weight ω was randomly drawn prior to training for

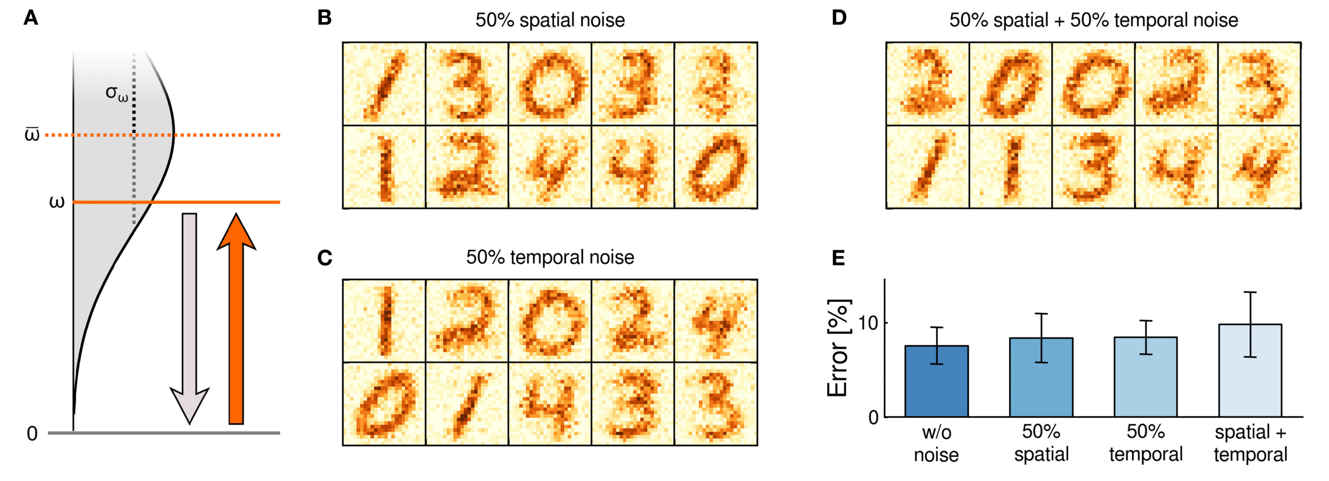


FIGURE 5 | Robustness to spatial and temporal noise. (A) Noise was added by drawing the weight ω of the active state from a normal distribution. Three types of noise were tested: For spatial noise, a value $\omega \sim \mathcal{N}(\omega; \bar{\omega}, \sigma_\omega^2)$ was assigned to each switch and maintained throughout the experiment. For temporal noise, the value ω was redrawn after every LTP transition. For spatial + temporal noise, the device specific spatial weight value determined the mean for redrawing ω after LTP transitions. **(B–D)** Number m_{ki} of active switches after learning at $t = 5000$ s. Shown are example networks for all three noise types. **(E)** Classification error after learning, based on 20 independently trained networks per noise type. Errorbars denote standard deviation among networks.

each stochastic switch from a normal distribution $\mathcal{N}(\omega\,;\,\bar{\omega}, \sigma_\omega^2)$ with mean $\bar{\omega}$ and standard deviation σ_ω. To capture the effect of temporal noise, in contrast, the weight ω was redrawn from $\mathcal{N}(\omega\,;\,\bar{\omega}, \sigma_\omega^2)$ whenever the constituent switched to its active state in an LTP transition. Furthermore, we examined the combined effect of both noise types being present simultaneously. In this combined case, the mean value for temporal noise was determined by the device-specific spatially perturbed weight value of each constituent. In any case, the range of perturbed weights was truncated to $\omega \geq 0$ to rule out negative conductances.

We repeated the experiment of **Figure 3** under each of these noise conditions. The mean $\bar{\omega} = 0.1$ was set to the undisturbed weight value of the previous, idealized experiment. The noise level was set to $\sigma_\omega = 0.05$, i.e., to 50% of the mean. Example cases of weight matrices after learning (shown are the m_{ki}'s from individual simulation runs) are presented in **Figures 5B–D** for the three noise conditions "spatial," "temporal" and "spatial+temporal," respectively. Surprisingly, hardly any difference to the idealized setup is observable. Nevertheless, under 20 repetitions of the learning simulation the detrimental influence of noise becomes visible in the classification performance (see **Figure 5E**) as noise appears to slightly increase the mean of the classification error. In summary, these results reveal a remarkable robustness of learning with compound memristive synapses to substantial device variability and severe temporal instability.

We next turned to the question how distorted switching probabilities π_{up} and π_{down} influence the learning dynamics in the WTA network. In the theory section, we had assumed that $\pi_{\text{up}} = \pi_{\text{down}}$, i.e., that the switching probabilities underlying LTP and LTD are balanced. This assumption, which could to some extent be achieved in a calibration step, yielded the elegant learning rule (3) and thereby facilitated the theoretical analysis. A physical implementation, however, will likely exhibit unbalanced switching probabilities in the majority of memristors. We examined the influence of unbalanced switching, in two ways. First, we applied spatial noise to the switching probabilities of individual constituents by drawing π_{up} and π_{down} (separately) from normal distributions with 50% noise level (truncated to $0 \leq \pi_{\text{up}}, \pi_{\text{down}} \leq 1$). Thus, about half of the stochastic switches were more responsive to LTP pulses, the other half more to LTD pulses; even more, some of the constituents only showed switching in one direction, or were completely unresponsive. Nevertheless, synapses developed a faithful representation of prototypic digits in a repetition of the experiment in **Figure 3** (data not shown). Also the classification performance was only mildly impaired (*classification error*: 8.7 ± 2.7% based on 20 networks) compared to ideal, noisefree synapses (*classification error*: 7.5 ± 1.9%).

In a second step, we pursued a slightly different–more principled–approach that permits a theoretical interpretation of how the altered synapse dynamics give rise to a different encoding of the expected input by the synaptic weights. Instead of drawing random parameters for each constituent, we systematically chose the LTD switching probability π_{down} larger (or smaller) than the LTP probability π_{up} throughout the synapse array. This systematic imbalance displays a worst-case scenario during learning since all synapses either favor (or suppress) LTD over LTP. We denote the relative imbalance between π_{up} and π_{down} by $\Delta := (\pi_{\text{up}} - \pi_{\text{down}})/\pi_{\text{up}}$. For instance, $\Delta = -0.5$ means that the probability for LTD transitions is 50% higher than for LTP transitions. **Figure 6A** shows examples of weight matrices after 5000 s of learning in a repetition of the experiment in **Figure 3**. In the top row, $\Delta = +0.5$, LTP transitions are favored over LTD transitions, resulting in generally stronger weights in comparison with balanced STDP ($\Delta = 0.0$, middle row). Conversely, in the bottom row, $\Delta = -0.5$, the systematic strengthening of LTD

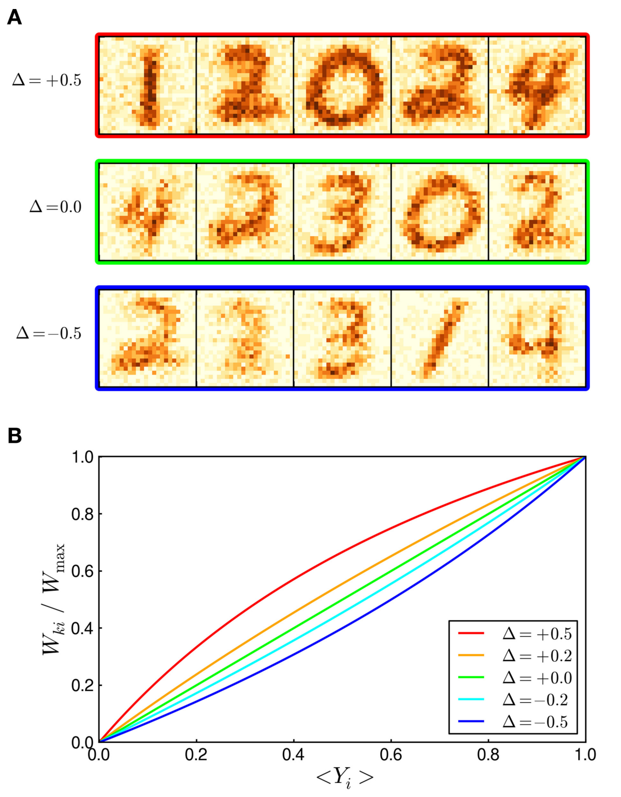


FIGURE 6 | Robustness to unbalanced switching probabilities. (A) Examples of learned weight matrices when the switching probabilities π_{up} and π_{down} are systematically unbalanced. The parameter $\Delta := (\pi_{\text{up}} - \pi_{\text{down}})/\pi_{\text{up}}$ measures the relative imbalance between LTP and LTD transitions. The top (bottom) row shows weight matrices resulting from a 50% decrease (increase) of the LTD switching probability. The balanced case is shown in the middle row for comparison. **(B)** Systematic imbalance ($\Delta \neq 0$) leads to changed equilibrium points in the STDP rule and can be theoretically understood as a non-linear encoding of the expected input by the synaptic weights.

leads to weaker weight patterns. Nevertheless, synaptic weight values converged to a dynamic equilibrium in either case since the STDP rule preserves its general stabilizing weight dependence. As can be expected from the prototypic digits that emerged in the weight matrices, the classification performance of the WTA networks was not considerably impaired by the unbalanced switching (classification errors estimated from 20 networks: $6.7 \pm 0.8\%$ for $\Delta = +0.5$; $7.5 \pm 1.9\%$ for $\Delta = 0$; $11.8 \pm 4.0\%$ for $\Delta = -0.5$). Indeed, positive Δ-values even performed slightly (but not significantly) better than balanced switching. A conceptual understanding of the altered learning dynamics can be obtained from the equilibrium points of the unbalanced STDP rule. The short calculation, that had led to Equation (8) for the balanced case, can be repeated for unbalanced switching probabilities. **Figure 6B** shows the resulting encoding of the expected input value $\langle Y_i \rangle_{p(Y_i \mid Z_k=1)}$ by the synaptic weight W_{ki} for different Δ-values. The example digits shown in panel A correspond to the red, green and blue graph in panel B, respectively. This analysis illustrates how unbalanced switching probabilities give rise to a non-linear encoding of the input in the WTA network. In particular, it can be seen how the same expected input value $\langle Y_i \rangle_{p(Y_i \mid Z_k=1)}$ leads to stronger weights for $\Delta > 0$, and weaker weights for $\Delta < 0$.

3. DISCUSSION

We have proposed the compound memristive synapse model for neuromorphic architectures that employs multiple memristors in parallel to form a plastic synapse. A fundamental property of the synapse model is that individual memristors exhibit stochastic switching between two stable memristive states rather than obeying a deterministic update rule. Yet, the expected weight change of the compound memristive synapse, as it arises from the stochastic switching of its constituents, yielded an STDP-type plasticity rule with a stabilizing, linear weight dependence. We examined the computational capabilities of the compound-synapse STDP rule in WTA networks, a common circuit motif in cortical and neuromorphic architectures, by analyzing the network and synapse dynamics from the perspective of probability theory and machine learning. The comprehensive mathematical treatment revealed that compound memristive synapses enable a spiking network to perform Bayesian inference in and autonomous statistical optimization of a Mixture-of-Gaussians generative model via generalized Expectation-Maximization. Accompanying computer simulations demonstrated the practical capability of the synapse model to perform unsupervised classification tasks and, furthermore, revealed a remarkable robustness of the compound synapses to substantial device variations and imperfections.

3.1. COMPREHENSIVE LEARNING THEORY OF MEMRISTIVE PLASTICITY

Our work contributes a theoretical foundation for memristive learning in neural networks to the endeavor to employ memristors as plastic synapses in self-calibrating systems. Snider (2008); Querlioz et al. (2011), and Serrano-Gotarredona et al. (2013) have investigated how different pre- and postsynaptic waveforms can shape a memristive STDP learning window. Jo et al. (2010) and Mayr et al. (2012) have demonstrated STDP-type plasticity in Ag/Si and BiFeO3 memristors. Yu et al. (2013) reported stochastic switching between stable states in oxide-based memristive synapses. Gaba et al. (2013) studied the parameter dependence of switching probabilities in metal filament based memristors, indicating a renewal process that is independent of the overall network firing rate. A strategy for integrating nanoscale memristive synapses into a hybrid memristor-CMOS network architecture was proposed by Indiveri et al. (2013). The beneficial contribution of stochasticity to learning with CMOS synapse circuits was explored by Chicca et al. (2014). Here we have established a firm link between the emergent synapse configurations observed in such architectures (see e.g., Querlioz et al., 2011) and a rigorous mathematical description of memristive learning on the system level using machine learning theory. Our findings on memristive learning from a Bayesian perspective build upon a series of theoretical contributions on synaptic learning in spiking neural networks: Nessler et al. (2009, 2013) identified a general link between STDP-type synaptic plasticity and statistical model optimization for probabilistic inference in WTA networks. Habenschuss et al. (2012) extended this work to incorporate also homeostatic intrinsic plasticity, thereby overcoming several limiting assumptions on the input presentation. Finally, Habenschuss et al. (2013) investigated how the learning framework can be generalized to support a broad class of probability distributions. We expect that utilizing machine learning theory for describing the effects of specific memristor synapse models can significantly promote our understanding of memristive learning and its computational prospects.

3.2. HEADING FOR A FULL HARDWARE INTEGRATION

Plasticity in the compound memristor synapse model relies on stochastic transitions between two stable states. Such bistable devices–or more generally, devices with a clearly discrete state spectrum–were reported to exhibit a high degree of uniformity (Lee et al., 2006a; Fang et al., 2011) and temporal stability (Indiveri et al., 2013). Notably, the theoretical approach we have persued in this work could likely be extended to cover memristors with more than two stable states and to support more complex input and plasticity mechanisms. For instance, the Gaussian likelihood distributions $p(\boldsymbol{Y} \mid \boldsymbol{Z})$ identified in the present study, in principle support inference over arbitrary real-valued input states $\boldsymbol{y}(t)$. Such states could arise if the input is presented in the form of exponentially decaying or additive postsynaptic potentials. Such more complex input types could afford more versatile STDP pulsing schemes, and the resulting memristor plasticity rules could likely be incorporated in an adapted model of statistical learning. The reason we restricted the input to binary values $y_i(t)$ is found in the STDP pulsing scheme that employs binary presynaptic waveforms. In this case, the theoretically derived generative model reveals how active and inactive inputs contribute to the network's spike response by means of a Gaussian likelihood distribution $p(Y_i = y_i(t) \mid Z_k = 1)$ that is sampled only at $y_i(t) = 0$ and $y_i(t) = 1$.

In this article, we have employed a simple model for stochastic switching in memristive devices where switching occurs with probabilities $\pi_{\text{up}}, \pi_{\text{down}}$ which depend on the applied voltage difference across the memristor terminals. This phenomenological model captures the most salient aspects of switching in real memristive materials (Jo et al., 2009b; Gaba et al., 2013) and was used as an abstraction of memristive switching in a recent experimental study (Suri et al., 2013). In future research, it will be important to evaluate the effectiveness of this model either with physical memristors or in simulations based on detailed memristor models. The authors of Suri et al. (2013) raised the concern that the precise switching probabilities of individual devices are potentially hard to control in large-scale systems. In this regard, our simulation results indicate that learning with compound memristive synapses tolerates significant noise levels in the switching probabilities. We expect the origin for the observed robustness to be twofold: Firstly, imbalances in $\pi_{\text{up}}, \pi_{\text{down}}$ between different constituents are expected to partly average out in compound synapses according to the central limit theorem; secondly, the stabilizing weight dependence of compound-synapse STDP ensures that even unbalanced switching leads to stable weight configurations, albeit with slightly shifted convergence points.

Another potential issue for learning with compound memristive synapses is the absolute value of the switching probability. The product $\pi_{\text{up}} \cdot W_{\max}$ can be linked to a learning rate in the theory domain (see **Table 1**) which controls how many samples from the input history are integrated into the implicit generative model during online learning. A slow and gradual memory formation, which is desirable for developing a representation of large and complex input data sets, relies on small learning rates, i.e., on small switching probabilities. It has to be seen if memristive materials that exhibit stochastic switching provide sufficiently small switching probabilities. A possible remedy in a hardware integration could be to multiplex the back-propagating signals from network neurons such that only a random subset of the memristors is notified of a network spike at a time (Fusi, 2002).

Regarding the physical model neurons of a hybrid memristor-CMOS architecture, two types of currents occur in the WTA network. The input integration via forward-synapses is spike based and could be realized with standard leaky integrators (see Methods). Lateral inhibition, in contrast, depends on the neuronal membrane potentials, and the involved inhibitory circuits should ideally transmit potentials instead of spikes. Alternatively, the effect of lateral inhibition could be approximated in a spike-based manner by populations of inhibitory neurons. Independent of the specific implementation of lateral inhibition, the resulting potential $u_k - u_{\text{inh}}$ controls the stochastic response of the WTA neurons that could either be implemented genuinely with a stochastic firing mechanism or be emulated with integrate-and-fire neurons (Petrovici et al., 2013).

3.3. INHERENTLY STOCHASTIC NATURE OF COMPOUND-SYNAPSE STDP

The spiking WTA network architecture with compound memristive synapses exploits stochasticity in various ways, in that the stochastic firing of network neurons in response to a transient input trajectory triggers stochastic STDP updates in the synaptic weights. From a learning perspective, the high degree of stochasticity contributes to the network's ongoing exploration for potential improvements in the parameter space. While the learning theory only guarantees convergence to a local optimum of the weight configuration, the stochastic nature of the ongoing exploration enables the network to evade small local optima in the parameter landscape, and thereby improves the robustness of learning (compared to traditional batch Expectation-Maximization).

For the derivation of the learning algorithm, we have focused on the weight-dependent STDP rule (3) which describes the expected temporal weight change $\langle \frac{d}{dt} W_{ki} \rangle$ of the compound synapse. The stochasticity of memristive switching, however, gives rise to a probability distribution over the weights, as well.

Table 1 | Correspondence of synapse parameters between the hardware and theory domain.

Parameter name	Hardware	Theory
Learning rate	$\pi_{\text{up}} \cdot W_{\max}$	η_W
Max. weight $W_{\max}$	$\omega \cdot M$	$1/\sigma^2$
Likelihood mean μ_{ki}	m_{ki}/M	$\sigma^2 \cdot W_{ki}$
Synaptic resolution	ω	$1/(M \cdot \sigma^2)$

Indeed, in equilibrium we expect that the number of active constituents m_{ki} follows a binomial-type weight distribution. This points to a potential knob for adjusting the amount of stochasticity used during online learning: when many memristors are recruited per synapse, i.e., for large M, we expect a reduced variance in the weight distribution.

Besides the level of stochasticity, the parameter M also controls the weight resolution of the compound synapse. In **Figure 4**, we have investigated the impact of the weight resolution on the learning capabilities of the WTA network. Notably, we observed that even with $M = 4$, i.e., with synapses that feature only 5 weight levels, the network performed reasonably well in the handwritten digit recognition task. The observation that even a low synaptic weight resolution can yield a satisfactory performance has important practical implications for nanoscale circuit designs, where integration density and power consumption impose crucial constraints, since the area allocated by the synapse array grows linearly in the synaptic size M. For instance, SRAM cells can be fabricated with a cell size of $0.127\,\mu\text{m}^2$ (Wu et al., 2009), corresponding to a memory density of $\lesssim 1\,\text{Gb/cm}^2$. Importantly, this estimate does not include any additional plasticity circuits for implementing STDP or similar plasticity mechanisms. Functional Ag/Si memristive crossbars with $2\,\text{Gb/cm}^2$ memory density were demonstrated by Jo et al. (2009a), with densities up to $10\,\text{Gb/cm}^2$ being envisioned (Jo et al., 2009a,b). In the long term, memristive crossbars are expected to combine the advantages of SRAM and Flash memory regarding energy efficiency, non-volatility and integration density (Yang et al., 2013).

3.4. GENERALIZATION TO OTHER MATERIALS AND FUTURE RESEARCH

In recent years, a plethora of (in a broader sense) memristive materials has been discovered, and the characterization and refinement of their switching dynamics is evolving rapidly. At least four types of stochastically switching memristive devices can be distinguished: Switching in (1) anion-based (e.g., HfO_x/TiO_x Yu et al., 2013) and (2) cation-based (e.g., Ag/GeS_2 Suri et al., 2013) devices mainly originates from conductive filament formation (Yang et al., 2013). In contrast, (3) single-electron latching switches [e.g., CMOS/MOLecular (CMOL) CrossNets Lee et al., 2006b] rely on electronic tunneling effects and, thus, their stochastic switching dynamics arise directly from the underlying physical process. Similarly, (4) magnetoresistive devices (e.g., spin-transfer torque magnetic memory (STT-MRAM) Vincent et al., 2014) can inherit stochastic switching dynamics from fundamental physical properties. Some manufacturing processes related to these ideas (like conductive-bridging RAM and STT-MRAM) reached already an early industrial stage, others are still primarily subject of academic research. While the microscopic origins of plasticity in these memristor types are fundamentally different, they all share stochastic, persistent switching between bistable memory states on a phenomenological level. We therefore believe that the compound memristive synapse model displays a promising concept for future work in diverse research fields.

Independent of the underlying switching mechanism, any nanoscale synaptic crossbar will likely exhibit imperfections and imbalances due to process variations. Here, we have investigated spatial and temporal noise in the weight values as well as deviations in the switching probabilities under unbiased, uniform conditions. However, physical implementations can be expected to also suffer from more systematic imperfections, such as structural imbalances (e.g., one corner of the array being more reactive) or crosstalk between neighboring devices. While our computer simulations indicate a remarkable general robustness against device variations of various types, additional research is required to estimate the influence of such systematic, and potentially coupled, deviations.

3.5. CONCLUSIONS

In this article, we have introduced the compound memristive synapse model together with the compound-synapse STDP rule for weight adaptation. Compound-synapse STDP, a stabilizing weight-dependent plasticity rule, naturally emerges under a standard STDP pulsing scheme. In addition, by employing memristors with bistable memristive states, compound memristive synapses may circumvent practical challenges in the design of reliable nanoscale memristive materials. Both, our theoretical analysis and our computer simulations confirmed that compound-synapse STDP endows networks of spiking neurons with powerful learning capabilities. Hence, the compound memristive synapse model may provide a synaptic design principle for future neuromorphic hardware architectures.

4. METHODS

4.1. PROBABILISTIC MODEL DEFINITION

The probabilistic model that corresponds to the spiking network is a mixture model with K mixture components and Gaussian likelihood function. Formally, we define a joint distribution $p(\boldsymbol{Y} = \boldsymbol{y}(t),\ \boldsymbol{Z} = \boldsymbol{z}(t)\,|\,\boldsymbol{\theta})$ over K hidden binary random variables $\boldsymbol{Z} = (Z_1, \ldots, Z_K)^\top$ with values $z_k(t) \in \{0, 1\}$, and N real-valued visible random variables $\boldsymbol{Y} = (Y_1, \ldots, Y_N)^\top$ with values $y_i(t) \in \mathbb{R}$. The parameter set $\boldsymbol{\theta} = \{\hat{\boldsymbol{b}},\ \boldsymbol{W}\}$ consists of a real-valued bias vector $\hat{\boldsymbol{b}} = (\hat{b}_1, \ldots, \hat{b}_K)^\top$ and a real-valued $K \times N$ weight matrix $\boldsymbol{W}$. The hidden RVs Z_k display an unrolled representation of a multinomial RV $\tilde{Z} \in \{1, \ldots, K\}$ that enumerates the mixture components, and we identify $\tilde{Z} = k \Leftrightarrow Z_k = 1$, i.e., exactly one binary RV Z_k is active in the random vector $\boldsymbol{Z}$. In the following, we stick to the unrolled vector notation $\boldsymbol{Z}$, and, for readability, omit the time-dependent notation and further shorten the notation by identifying the RVs with their values, e.g., we write $p(\boldsymbol{z}\,|\,\boldsymbol{\theta})$ for $p(\boldsymbol{Z} = \boldsymbol{z}(t)\,|\,\boldsymbol{\theta})$.

The likelihood distribution fulfills the naïve Bayes property, i.e., all statistical dependencies between visible RVs y_i, y_j are explained by the hidden state $\boldsymbol{z}$. More precisely, the generative model has the following structure:

$$p(\boldsymbol{y}, \boldsymbol{z}\,|\,\boldsymbol{\theta}) = p(\boldsymbol{z}\,|\,\boldsymbol{\theta}) \cdot \prod_{k=1}^{K} \prod_{i=1}^{N} p\left(y_i\,|\,z_k = 1, \boldsymbol{\theta}\right)^{z_k} \quad , \tag{12}$$

with the prior

$$p(\boldsymbol{z}\,|\,\boldsymbol{\theta}) = \frac{e^{\boldsymbol{z}^\top \hat{\boldsymbol{b}}}}{\sum_{j=1}^{K} e^{\hat{b}_j}} \tag{13}$$

and the likelihood

$$p\left(y_i \mid z_k = 1, \boldsymbol{\theta}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(y_i-\mu_{ki})^2}{2\sigma^2}}\,, \tag{14}$$

with σ^2 denoting an arbitrary (but fixed) variance which displays a constant in the model. Equation (14) defines a Gaussian likelihood model for each input RV y_i with mean μ_{ki} which is selected by the active hidden cause $z_k = 1$ in Equation (12). For theoretical considerations, it is convenient to reorganize Equation (14) according to its dependency structure:

$$p\left(y_i \mid z_k = 1, \boldsymbol{\theta}\right) = \frac{e^{-y_i^2/(2\sigma^2)}}{\sqrt{2\pi\sigma^2}} \cdot e^{\frac{\mu_{ki}}{\sigma^2}\cdot y_i} \cdot e^{-\mu_{ki}^2/(2\sigma^2)} \tag{15}$$

$$=: h(y_i) \cdot e^{W_{ki}\cdot y_i} \cdot e^{-A_{ki}}\,, \tag{16}$$

where we set $W_{ki} = \mu_{ki}/\sigma^2$ and $A_{ki} = \mu_{ki}^2/(2\sigma^2) = \sigma^2 W_{ki}^2/2$. The first factor does neither depend on the hidden causes z_k nor on the weight W_{ki} and will play no role during inference and learning. The second factor describes the coupling between the visible RV y_i and the active latent RV z_k through the mean $W_{ki} = \mu_{ki}/\sigma^2$. Finally, the third factor solely depends on the weight W_{ki} (and not on y_i) and ensures correct normalization of the distribution.

4.2. INFERENCE

The posterior distribution given an observation $\boldsymbol{y}$ follows directly from Bayes rule:

$$p\left(\boldsymbol{z} \mid \boldsymbol{y}, \boldsymbol{\theta}\right) = p(\boldsymbol{z} \mid \boldsymbol{\theta}) \cdot p\left(\boldsymbol{y} \mid \boldsymbol{z}, \boldsymbol{\theta}\right) \;/\; \text{Norm.} \tag{17}$$

$$= e^{\boldsymbol{z}^{\mathsf{T}}\hat{\boldsymbol{b}}} \prod_{k=1}^{K}\prod_{i=1}^{N} h(y_i)^{z_k} \cdot$$

$$e^{z_k\cdot W_{ki}\cdot y_i} \cdot e^{-z_k\cdot A_{ki}} \;/\; \text{Norm.} \tag{18}$$

$$= e^{\boldsymbol{z}^{\mathsf{T}}\cdot\left[\hat{\boldsymbol{b}} - \boldsymbol{A} + \boldsymbol{W}\cdot\boldsymbol{y}\right]} \cdot \prod_{i=1}^{N} h(y_i) \;/\; \text{Norm.} \tag{19}$$

with $\boldsymbol{A} := (A_1, \ldots, A_k, \ldots, A_K)^{\mathsf{T}}$ and $A_k := \sum_{i=1}^{N} A_{ki}$. We evaluate the posterior for a specific hidden RV z_k to be active and provide the normalization constant explicitly:

$$p\left(z_k = 1 \mid \boldsymbol{y}, \boldsymbol{\theta}\right) = \frac{e^{\hat{b}_k - A_k + \sum_{i=1}^{N} W_{ki}\cdot y_i} \cdot \prod_{i=1}^{N} h(y_i)}{\sum_{j=1}^{K} e^{\hat{b}_j - A_j + \sum_{i=1}^{N} W_{ji}\cdot y_i} \cdot \prod_{i=1}^{N} h(y_i)} \tag{20}$$

$$= \frac{e^{\hat{u}_k}}{\sum_{j=1}^{K} e^{\hat{u}_j}} \tag{21}$$

where we defined $\hat{u}_k = \hat{b}_k - A_k + \sum_{i=1}^{N} W_{ki}\cdot y_i$. The quantities $\hat{u}_k$ are reminiscent of neuronal membrane potentials which consist of bias terms $\hat{b}_k - A_k$ and synaptic input $\sum_{i=1}^{N} W_{ki}\, y_i$. However, implicitly the bias terms depend on all afferent synaptic weights since $\hat{b}_k - A_k = \hat{b}_k - \frac{\sigma^2}{2}\sum_i W_{ki}^2$ and, thus, rely on information not locally available to the neurons. This issue will be resolved in the context of learning: We will identify update rules for both biases and synapses which only use information available locally and thereby make a neural network implementation feasible.

4.3. LEARNING VIA GENERALIZED EXPECTATION-MAXIMIZATION

We investigate unsupervised learning of the probabilistic model based on generalized online Expectation-Maximization (EM) (Dempster et al., 1977), an optimization algorithm from machine learning theory. To this end, we impose additional constraints on the posterior distribution (Graça et al., 2007) which will enable a neural network implementation via homeostatic intrinsic plasticity (Habenschuss et al., 2012) and STDP-type synaptic plasticity (Habenschuss et al., 2013; Nessler et al., 2013). Since the derivation is almost identical to Habenschuss et al. (2012), we only outline the key steps and main results in the following and refer to Habenschuss et al. (2012) for a the details.

The algorithmic approach rests upon the generalized EM decomposition:

$$\mathcal{F}(\boldsymbol{W}, q(\boldsymbol{z}|\boldsymbol{y})) = \mathcal{L}(\boldsymbol{\theta}) - \left\langle \mathrm{D_{KL}}\left(q(\boldsymbol{z}|\boldsymbol{y}) \,||\, p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{\theta})\right)\right\rangle_{p^*(\boldsymbol{y})}$$

$$\to \text{E-step} \tag{22}$$

$$= \langle \log p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) \rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})} + \langle H(q(\boldsymbol{z}|\boldsymbol{y})) \rangle_{p^*(\boldsymbol{y})}$$

$$\to \text{M-step} \tag{23}$$

with the log-likelihood $\mathcal{L}(\boldsymbol{\theta}) = \langle \log p(\boldsymbol{y} \mid \boldsymbol{\theta}) \rangle_{p^*(\boldsymbol{y})}$, the Kullback-Leibler divergence $\mathrm{D_{KL}}\left(q(\boldsymbol{z}) \,||\, p(\boldsymbol{z})\right) = \sum_{\boldsymbol{z}} q(\boldsymbol{z}) \cdot \log\left(q(\boldsymbol{z})/p(\boldsymbol{z})\right)$ and the entropy $H(q(\boldsymbol{z})) = -\sum_{\boldsymbol{z}} q(\boldsymbol{z}) \cdot \log q(\boldsymbol{z})$. The distribution $p^*(\boldsymbol{y})$ denotes the input distribution actually presented to the system. The distribution $p(\,\cdot \mid \boldsymbol{\theta})$ is the probabilistic model defined above. The distribution $q(\boldsymbol{z}|\boldsymbol{y})$ is called variational posterior and will ultimately be implemented by the spiking network. The short hand notation $\langle\,\cdot\,\rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})}$ denotes the concatenated average $\langle\,\langle\,\cdot\,\rangle_{q(\boldsymbol{z}|\boldsymbol{y})}\,\rangle_{p^*(\boldsymbol{y})}$ with respect to the input distribution and the resulting variational posterior. In principle, the above decomposition holds for any choice of q, and since the Kullback-Leibler divergence in Equation (22) is strictly non-negative, the objective function $\mathcal{F}$ is a lower bound of the log-likelihood $\mathcal{L}$. During optimization the algorithm will persue two goals: to increase $\mathcal{L}$, i.e., to better adapt the probabilistic model to the data, and to keep $\langle\, \mathrm{D_{KL}}\left(q \,||\, p\right)\rangle$ small, i.e., to maintain a reliable approximation $q(\boldsymbol{z}|\boldsymbol{y})$ of the exact posterior $p\left(\boldsymbol{z} \mid \boldsymbol{y}, \boldsymbol{\theta}\right)$.

We first impose a homeostatic constraint on the variational posterior $q(\boldsymbol{z}|\boldsymbol{y})$, namely that the long term average activation of any hidden RV z_k matches a predefined target value c_k (with $\sum_k c_k = 1$). Formally, we define a set of constrained distributions $\mathcal{Q} = \{q : \langle z_k \rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})} = c_k \;\; \forall\, 1 \le k \le K\}$ and demand $q(\boldsymbol{z}|\boldsymbol{y}) \in \mathcal{Q}$. The optimization algorithm then relies on the joint application of an E(expectation)-step and an M(aximization)-step: During the E-step, we aim to minimize the Kullback-Leibler divergence with respect to $q \in \mathcal{Q}$ in Equation (22); during the M-step, we perform gradient ascent on $\langle \log p(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{\theta}) \rangle$ with respect to the weights W_{ki} in Equation (23). The E- and M-step will be discussed separately.

The E-step is a constrained optimization problem, namely the minimization of $\langle\, \mathrm{D_{KL}}\left(q \,||\, p\right)\rangle$ such that $q \in \mathcal{Q}$, that can be solved through Lagrange multipliers. Since we imposed K constraints (one per RV z_k), we need K Lagrange multipliers β_k. It turns out that the solution to this optimization problem simply adds the multipliers β_k to the biases $\hat{b}_k - A_k$ in Equation (20). This convenient result gives rise to the definition of intrinsic excitabilities $b_k := \hat{b}_k - A_k + \beta_k$ which unify biases and multipliers in a single quantity. Furthermore, it turns out that the optimal values of the β_k's (and thus the b_k's) can be determined via iterative update rules that solely rely on the hidden RVs z_k under the variational response $q(\boldsymbol{z}|\boldsymbol{y})$ and overwrite the non-local terms A_k. In summary, we obtain the variational posterior distribution q that solves the E-step:

$$q(\boldsymbol{z}|\boldsymbol{y}) = \frac{e^{u_k}}{\sum_{j=1}^{K} e^{u_j}} \quad \text{with} \quad u_k = b_k + \sum_{i=1}^{N} W_{ki} \cdot y_i \tag{24}$$

$$\Delta b_k \propto \langle\, c_k - z_k \,\rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})} \quad . \tag{25}$$

The variational posterior in Equation (24) is described in terms of membrane potentials u_k which consist of synaptic input $\sum_i W_{ki}\, y_i$ and intrinsic excitabilities b_k. Equation (25) regulates the intrinsic excitabilities b_k in a homeostatic fashion: When the average response exceeds the target c_k, the excitability is reduced, and vice versa.

The M-step can be solved via gradient ascent on $\mathcal{F}$ with respect to the weights W_{ki}. The variational posterior q is a constant during the M-step in EM, and thus the log-joint distribution $\log p(\boldsymbol{y}, \boldsymbol{z}\,|\,\boldsymbol{\theta})$ remains as the only W_{ki}-dependent term in Equation (23). By taking the derivative of the log-joint defined by Equation (12), (13), and (16) with respect to W_{ki}, we obtain:

$$\frac{d\mathcal{F}}{dW_{ki}} = \langle\, \partial_{W_{ki}} \log p(\boldsymbol{y}, \boldsymbol{z}\,|\,\boldsymbol{\theta}) \,\rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})} \tag{26}$$

$$= \langle\, \partial_{W_{ki}} z_k \cdot \log p\left(y_i\,|\,z_k = 1, \boldsymbol{\theta}\right) \rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})} \tag{27}$$

$$= \langle\, z_k \cdot (y_i - \sigma^2 \cdot W_{ki}) \,\rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})} \quad . \tag{28}$$

The gradient with respect to the weights W_{ki} yields Hebbian-type update rules that use pre-(y_i) and post-(z_k) synaptic activity and the current weight W_{ki} given the input $p^*(\boldsymbol{y})$ and the variational response $q(\boldsymbol{z}|\boldsymbol{y})$. Importantly, only local information is required during the E- and M-step.

4.4. SPIKING NETWORK IMPLEMENTATION

The spiking neural network model instantiates Equation (24), (25), and (28), i.e., it represents the variational posterior $q(\boldsymbol{z}|\boldsymbol{y})$ for probabilistic inference through its spike response and implements the derived update rules for generalized online EM learning through intrinsic and synaptic plasticity.

Each of the RVs z_k is represented by one of the K network neurons, and each spike in the network is a sample from the variational posterior $q(\boldsymbol{z}|\boldsymbol{y})$ by identifying $z_k = 1$ for a spike of the k-th network neuron. By setting the instantaneous firing rate ρ_k to be

$$\rho_k = \lim_{\delta t \to 0} p(\text{spike in}[t, t + \delta t])/\delta t = r_{\text{net}} \cdot e^{u_k - u_{\text{inh}}} \tag{29}$$

with $u_{\text{inh}} := \log \sum_{j=1}^{K} \exp(u_j)$ the network thus implements Equation (24) for any choice of r_{net}, i.e., $p_{\text{net}} = q$.

The learning rules (25) and (28) rely on expected values $\langle \cdot \rangle_{p^*(\boldsymbol{y})q(\boldsymbol{z}|\boldsymbol{y})}$. The expectations can be approximated from input samples $\boldsymbol{y} \sim p^*(\boldsymbol{y})$ and posterior samples $\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{y})$ in response to this input. The input vector $\boldsymbol{y}$ is defined at any time t in the network as it measures the instantaneous presence or absence of rectangular input pulses. Samples of the latent variable $\boldsymbol{z}$, in contrast, are only defined at the spike times of the network. Hence integrating expected values $\langle\, z_k \,\rangle$ from the spike response can be expressed most conveniently in terms of the spike train function $s_k(t) = \sum_f \delta(t - t_k^f)$ of the network neurons. We obtain the following plasticity rules:

$$\frac{db_k}{dt} = \eta_b \cdot [r_{\text{net}} \cdot c_k - s_k(t)] \tag{30}$$

$$\frac{dW_{ki}}{dt} = \eta_W \cdot s_k(t) \cdot \left[y_i - \sigma^2 \cdot W_{ki}\right] \tag{31}$$

with small learning rates η_b and η_W. The homeostatic rule (30) regulates the intrinsic excitabilities b_k such that the average target activations $\langle\, z_k(t) \,\rangle \approx c_k$ are maintained over the presentation of many different input patterns $\boldsymbol{y}(t) \sim p^*(\boldsymbol{y})$ in accordance with Equation (25), and thereby implements the E-step. Building on the network response shaped by the E-step, the synaptic rule (31) on average increases the objective function $\mathcal{F}$ since synaptic changes $\frac{d}{dt} W_{ki}$ on average point in the direction of the $\boldsymbol{W}$-gradient of $\mathcal{F}$ given by Equation (28), thereby implementing the M-step. Since synaptic updates rely on a (sufficiently) precise E-step which, in turn, needs to integrate any changes in the network response due to synaptic plasticity, homeostatic intrinsic plasticity is required to act on faster time scales than synaptic plasticity. As a consequence, the learning rate η_b will typically exceed the learning rate η_W in the spiking network implementation.

The homeostatic intrinsic plasticity rule (30) can readily be implemented by the spiking neurons: The intrinsic excitability b_k of each neuron increases linearly in time with a slow drift $\eta_b \cdot r_{\text{net}} \cdot c_k$ and is lowered abruptly by η_b at the spike times of neuron z_k. Similarly, mapping the synaptic plasticity rule (31) to the compound-synapse STDP rule (3) is straight-forward due to the structural equivalence of Equations (3) and (31): The learning rate η_W in the theory domain corresponds to $\pi_{\text{up}} \cdot W_{\max}$ in the hardware domain, e.g., high jumping probabilities π_{up} and large weight contributions $\omega = W_{\max}/M$ of individual stochastic switches lead to high learning rates η_W. Furthermore, the maximum weight $W_{\max}$ can directly be identified with the precision $1/\sigma^2$ of the likelihood distribution (14). In the theory domain, we know that $\mu_{ki} = \sigma^2 \cdot W_{ki}$, and hence, $\mu_{ki} = W_{ki}/W_{\max} = m_{ki}/M$ for the compound synapses. Finally, due to the structural equivalence of Equations (3) and (31) we find that the compound memristor plasticity rule (3) inherits the convergence properties from the theoretically derived plasticity rule (31) during online learning. The resulting translation of memristor synapse parameters to the abstract model is summarized in **Table 1**.

4.5. COMPUTER SIMULATIONS

All computer simulations were performed with customized Python scripts. For the computer simulations, we employed a network architecture with $K = 10$ network neurons and $N = 24 \cdot 24 = 576$ inputs. The simulation time step was $\delta t = 1$ ms and the PSP time constant $\tau = 10$ ms. The overall network firing rate was set to $r_{\text{net}} = 100$ Hz, the homeostatic target activation uniformly to $c_k = 1/K$. Synapses were composed of $M = 10$ constituents with weight $\omega = 0.1$ each. Switching probabilities were set to $\pi_{\text{up}} = \pi_{\text{down}} = 10^{-3}$. This corresponds to a Gaussian likelihood model with variance $\sigma^2 = 1$ and learning rate $\eta_W = 10^{-3}$. The learning rate for homeostatic intrinsic plasticity was set to $\eta_b = 20 \cdot \eta_W$. For the simulations in **Figures 4–6**, certain parameters deviated from the above, depending on the simulation setup. For **Figure 6**, the switching probability $\pi_{\text{up}} = 10^{-3}$ was kept fixed, and π_{down} was adapted for different Δ-values. All other changes are described directly in the Results section.

For learning experiments, digits 0, 1, 2, 3, 4 were extracted in equal proportion from the MNIST training data set (LeCun et al., 1998). A frame of two pixels width was removed, leaving images of size 24 × 24. The images (indexed by s) were scaled linearly to activity patterns $x_i^s \in [0.05, 0.9]$, with $i = 1, \ldots, N$, which were presented to the network as follows. For given activity pattern $\boldsymbol{x}^s = (x_1^s, \ldots, x_N^s)$, each input i spiked with probability $p_i^{\text{spike}} = 1 - \left(1 - x_i^s\right)^{\delta t/\tau}$ per time step δt. During training, a new activity pattern $\boldsymbol{x}^s$ was randomly drawn from the training set every 100 ms. Each network was trained for 5000 s. To obtain the unweighted PSP values y_i, the resulting spike patterns were convolved with a box kernel of duration τ and amplitude 1, and then clipped to values [0, 1]. This defined the input $\boldsymbol{y}(t)$, and thus (implicitly) the data distribution $\boldsymbol{y}(t) \sim p^*(\boldsymbol{y})$. Notably, the spiking probability p_i^{spike} is chosen such that $\langle y_i \rangle = x_i^s$.

The estimate of the log-likelihood in **Figure 3D** was based on 5000 input samples $\boldsymbol{y}(t)$, which were randomly drawn from the training data, and assumed a uniform prior $p(z_k = 1 \,|\, \boldsymbol{\theta}) = 1/K$ in accordance with the homeostatic target activation. The classification performance in **Figures 3D**, **4B**, **5E** was determined as follows. For given configuration of the synapses and intrinsic excitabilities, 100 versions of each digit from the training data set were presented to the network for 1 s each. Each neuron was labeled to be tuned to the digit class it was most responsive to. Then 500 versions of each digit from the MNIST test data set were presented to the network for 1 s each. The network neuron that spiked most during the 1 s period determined the network's classification of the input digit. The classification error is the fraction of wrongly classified digits.

4.6. IMPLEMENTATION WITH LEAKY INTEGRATOR NEURONS

The idealized stochastic neurons in the WTA network model feature abstact membrane potentials u_k that integrate the input $\boldsymbol{y}(t)$ through the weights W_{ki} linearly. In a hardware integration, however, synaptic weights arise from the conductance of memristors, and neurons are physical implementations based on capacitors and various other circuit elements. Here, we outline one possible hardware integration and consider a leaky integrator with membrane potential U_k that obeys the following dynamics:

$$\tau_{\text{m}} \cdot \frac{dU_k}{dt} = -(U_k - B_k) + I_k/G_{\text{L}} \quad , \tag{32}$$

with membrane time constant τ_{m}, leak conductance G_{L}, resting potential B_k and synaptic input current I_k. In the setup of **Figure 1A**, input spikes trigger a rectangular voltage pulse of duration τ and with amplitude U_{pre}. Denoting the conductance of the memristive synapse by G, this generates a synaptic current $I = U_{\text{pre}} \cdot G$. The equilibrium membrane potential (i.e., $\frac{d}{dt} U_k = 0$) under this current is $U_k = B_k + (U_{\text{pre}}/G_{\text{L}}) \cdot G$. For small membrane time constant $\tau_{\text{m}} \to 0$, e.g., for a small neuron capacitance, the fast membrane will closely resemble the rectangular presynaptic pulse shape, and the PSP amplitude $(U_k - B_k)$ will be proportional to the weight G. This linear integration property of U_k also holds in case of multiple memristive synapses acting in parallel, and we find

$$U_k = B_k + \frac{U_{\text{pre}}}{G_{\text{L}}} \cdot \sum_i G_{ki} \cdot y_i \quad . \tag{33}$$

Consequently, the membrane potential U_k of the leaky integrator matches the idealized membrane potential u_k employed in the Results section up to a linear function that serves to translate the voltage-based potential U_k to the unitless potential u_k. Using the membrane potential U_k, the exponential firing behavior (29) of the neurons could either be realized with an inherently stochastic firing mechanism. Alternatively, deterministic leaky integrate-and-fire neurons could be operated in a stochastic regime by adapting, for instance, the approach taken in Petrovici et al. (2013).

AUTHOR CONTRIBUTIONS

Johannes Bill and Robert Legenstein conceived the study, designed the experiments and wrote the manuscript. Johannes Bill developed the theory and performed the computer simulations.

FUNDING

Written under partial support of CHIST-ERA ERA-Net (Project FWF #I753-N23, PNEUMA) (Johannes Bill, Robert Legenstein) and the European Union project #604102 The Human Brain Project (HBP) (Robert Legenstein).

ACKNOWLEDGMENTS

We thank Radu Berdan, Alex Serb and Themis Prodromakis for valuable discussions on memristors, Giacomo Indiveri for initial discussions on the compound memristive synapse model, Georg Goeri for preparatory studies on robustness to device variability, and David Kappel as well as two anonymous reviewers for helpful comments on the manuscript. Johannes Bill thanks Bernhard Nessler and Stefan Habenschuss for all the fruitful discussions on the SEM theory and statistical learning in general.

REFERENCES

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

Caporale, N., and Dan, Y. (2008). Spike timing-dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.* 31, 25–46. doi: 10.1146/annurev.neuro.31.060407.125639

Chicca, E., Stefanini, F., Bartolozzi, C., and Indiveri, G. (2014). Neuromorphic electronic circuits for building autonomous cognitive systems. *Proc. IEEE* 102, 1367–1388. doi: 10.1109/JPROC.2014.2313954

Choi, H., Jung, H., Lee, J., Yoon, J., Park, J., Seong, D.-J., et al. (2009). An electrically modifiable synapse array of resistive switching memory. *Nanotechnology* 20:345201. doi: 10.1088/0957-4484/20/34/345201

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *J. R. Stat. Soc. B* 39, 1–38.

Douglas, R. J., and Martin, K. A. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci.* 27, 419–451. doi: 10.1146/annurev.neuro.27.070203.144152

Fang, Z., Yu, H., Li, X., Singh, N., Lo, G., and Kwong, D. (2011). Hfox/tiox/hfox/tiox multilayer-based forming-free rram devices with excellent uniformity. *IEEE Elexctron Device Lett.* 32, 566–568. doi: 10.1109/LED.2011.2109033

Fusi, S. (2002). Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates. *Biol. Cybern.* 87, 459–470. doi: 10.1007/s00422-002-0356-8

Gaba, S., Sheridan, P., Zhou, J., Choi, S., and Lu, W. (2013). Stochastic memristive devices for computing and neuromorphic applications. *Nanoscale* 5, 5872–5878. doi: 10.1039/c3nr01176c

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511815706

Graça, J. V., Ganchev, K., and Taskar, B. (2007). "Expectation maximization and posterior constraints," in *Advances in Neural Information Processing Systems* (Vancouver).

Habenschuss, S., Bill, J., and Nessler, B. (2012). "Homeostatic plasticity in bayesian spiking networks as expectation maximization with posterior constraints," in *Advances in Neural Information Processing Systems* (Lake Tahoe), 773–781.

Habenschuss, S., Puhr, H., and Maass, W. (2013). Emergence of optimal decoding of population codes through stdp. *Neural Comput.* 25, 1371–1407. doi: 10.1162/NECO-a-00446

Indiveri, G. (2000). Modeling selective attention using a neuromorphic analog vlsi device. *Neural Comput.* 12, 2857–2880. doi: 10.1162/089976600300014755

Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., and Prodromakis, T. (2013). Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology* 24:384010. doi: 10.1088/0957-4484/24/38/384010

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 1297–1301. doi: 10.1021/nl904092h

Jo, S. H., Kim, K.-H., and Lu, W. (2009a). High-density crossbar arrays based on a si memristive system. *Nano Lett.* 9, 870–874. doi: 10.1021/nl8037689

Jo, S. H., Kim, K.-H., and Lu, W. (2009b). Programmable resistance switching in nanoscale two-terminal devices. *Nano Lett.* 9, 496–500. doi: 10.1021/nl803669s

Jolivet, R., Rauch, A., Lüscher, H.-R., and Gerstner, W. (2006). Predicting spike timing of neocortical pyramidal neurons by simple threshold models. *J. Comput. Neurosci.* 21, 35–49. doi: 10.1007/s10827-006-7074-5

Kappel, D., Nessler, B., and Maass, W. (2014). Stdp installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLoS Comput. Biol.* 10:e1003511. doi: 10.1371/journal.pcbi.1003511

Keck, C., Savin, C., and Lücke, J. (2012). Feedforward inhibition and synaptic scaling–two sides of the same coin? *PLoS Comput. Biol.* 8:e1002432. doi: 10.1371/journal.pcbi.1002432

Kuzum, D., Jeyasingh, R. G., Lee, B., and Wong, H.-S. P. (2011). Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* 12, 2179–2186. doi: 10.1371/journal.pcbi.1002432

Lansner, A. (2009). Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations. *Trends Neurosci.* 32, 178–186. doi: 10.1016/j.tins.2008.12.002

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lee, D., Seong, D.-J., Jung Choi, H., Jo, I., Dong, R., Xiang, W., et al. (2006a). "Excellent uniformity and reproducible resistance switching characteristics of doped binary metal oxides for non-volatile resistance memory applications," in *Electron Devices Meeting, 2006. IEDM'06. International* (San Francisco: IEEE), 1–4.

Lee, J. H., Ma, X., and Likharev, K. (2006b). Cmol crossnets: possible neuromorphic nanoelectronic circuits. *Adv. Neural Inf. Process. Syst.* 18:755.

Malenka, R. C., and Bear, M. F. (2004). Ltp and ltd: an embarrassment of riches. *Neuron* 44, 5–21. doi: 10.1016/j.neuron.2004.09.012

Markram, H., Gerstner, W., and Sjöström, P. J. (2012). Spike-timing-dependent plasticity: a comprehensive overview. *Front. Synaptic Neurosci.* 4:2. doi: 10.3389/fnsyn.2012.00002

Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science* 275, 213–215. doi: 10.1126/science.275.5297.213

Mayr, C., Stärke, P., Partzsch, J., Cederstroem, L., Schüffny, R., Shuai, Y., et al. (2012). "Waveform driven plasticity in bifeo3 memristive devices: model and implementation," in *Advances in Neural Information Processing Systems* (Lake Tahoe), 1700–1708.

Mead, C., and Ismail, M. (1989). *Analog VLSI Implementation of Neural Systems*. Norwell, MA: Springer. doi: 10.1007/978-1-4613-1639-8

Morrison, A., Aertsen, A., and Diesmann, M. (2007). Spike-timing-dependent plasticity in balanced random networks. *Neural Comput.* 19, 1437–1467. doi: 10.1162/neco.2007.19.6.1437

Nessler, B., Pfeiffer, M., Buesing, L., and Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput. Biol.* 9:e1003037. doi: 10.1371/journal.pcbi.1003037

Nessler, B., Pfeiffer, M., and Maass, W. (2009). "Stdp enables spiking neurons to detect hidden causes of their inputs," in *Advances in Neural Information Processing Systems* (Vancouver), 1357–1365.

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., and Meier, K. (2013). Stochastic inference with deterministic spiking neurons. arXiv preprint arXiv:1311.3211.

Querlioz, D., Bichler, O., and Gamrat, C. (2011). "Simulation of a memristor-based spiking neural network immune to device variations," in *Neural Networks (IJCNN), The 2011 International Joint Conference on* (IEEE), 1775–1781.

Schemmel, J., Fieres, J., and Meier, K. (2008). "Wafer-scale integration of analog neural networks," in *Neural Networks, 2008. IJCNN 2008 (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on* (Hong Kong: IEEE), 431–438.

Serrano-Gotarredona, T., Masquelier, T., Prodromakis, T., Indiveri, G., and Linares-Barranco, B. (2013). Stdp and stdp variations with memristors for spiking neuromorphic learning systems. *Front. Neurosci.* 7:2. doi: 10.3389/fnins.2013.00002

Simoncelli, E. P., Paninski, L., Pillow, J., and Schwartz, O. (2004). Characterization of neural responses with stochastic stimuli. *Cogn. Neurosci.* 3, 327–338.

Snider, G. S. (2008). "Spike-timing-dependent learning in memristive nanodevices," in *Nanoscale Architectures, 2008. NANOARCH 2008. IEEE International Symposium* (Anaheim, CA: IEEE), 85–92.

Suri, M., Querlioz, D., Bichler, O., Palma, G., Vianello, E., Vuillaume, D., et al. (2013). Bio-inspired stochastic computing using binary cbram synapses. *Elect. Devices IEEE Trans.* 60, 2402–2409. doi: 10.1109/TED.2013.2263000

Van Rossum, M. C., Bi, G. Q., and Turrigiano, G. G. (2000). Stable hebbian learning from spike timing-dependent plasticity. *J. Neurosci.* 20, 8812–8821.

Vincent, A., Larroque, J., Zhao, W., Romdhane, N. B., Bichler, O., Gamrat, C., et al. (2014). "Spin-transfer torque magnetic memory as a stochastic memristive synapse," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium* (Melbourne: IEEE), 1074–1077.

Wu, S.-Y., Liaw, J., Lin, C., Chiang, M., Yang, C., Cheng, J., et al. (2009). "A highly manufacturable 28nm cmos low power platform technology with fully functional 64mb sram using dual/tripe gate oxide process," in *VLSI Technology, 2009 Symposium on* (IEEE), 210–211.

Yang, J. J., Strukov, D. B., and Stewart, D. R. (2013). Memristive devices for computing. *Nat. Nanotechnol.* 8, 13–24. doi: 10.1038/nnano.2012.240

Yu, S., Gao, B., Fang, Z., Yu, H., Kang, J., and Wong, H.-S. P. (2013). Stochastic learning in oxide binary synaptic device for neuromorphic computing. *Front. Neurosci.* 7:186. doi: 10.3389/fnins.2013.00186

Zamarreño-Ramos, C., Camuñas-Mesa, L. A., Pérez-Carrasco, J. A., Masquelier, T., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex. *Front. Neurosci.* 5:26. doi: 10.3389/fnins.2011.00026

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 30 September 2014; accepted: 24 November 2014; published online: 16 December 2014.

Citation: Bill J and Legenstein R (2014) A compound memristive synapse model for statistical learning through STDP in spiking neural networks. Front. Neurosci. ***8****:412. doi: 10.3389/fnins.2014.00412*

This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.