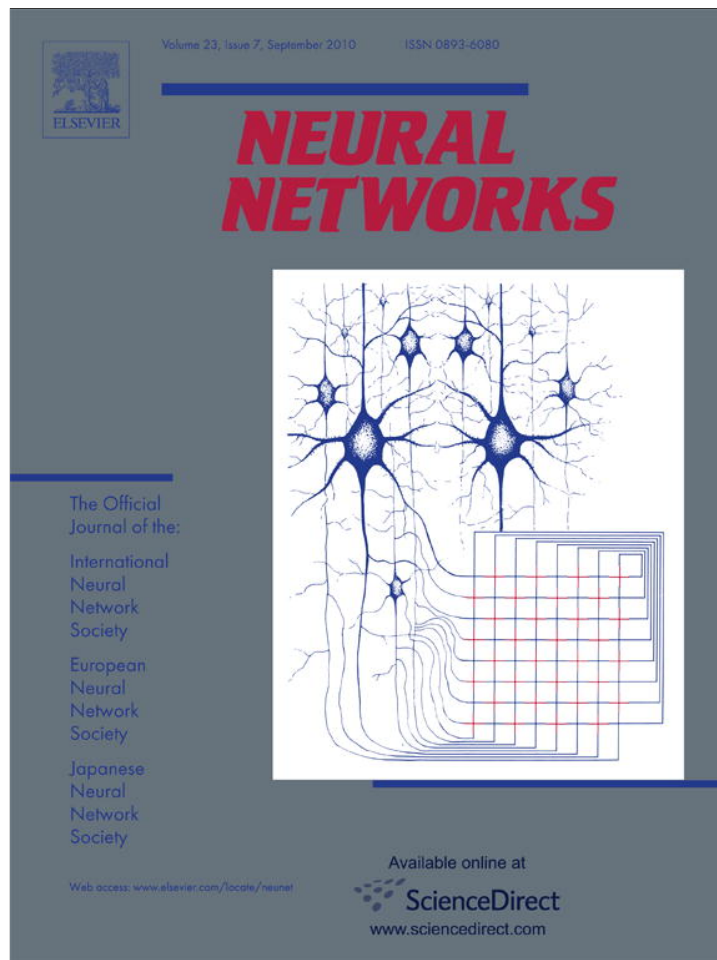


Provided for non-commercial research and education use.  
Not for reproduction, distribution or commercial use.



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

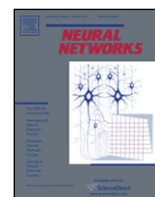
In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



Contents lists available at ScienceDirect

## Neural Networks

journal homepage: [www.elsevier.com/locate/neunet](http://www.elsevier.com/locate/neunet)

## PAX: A mixed hardware/software simulation platform for spiking neural networks

S. Renaud<sup>a,\*</sup>, J. Tomas<sup>a</sup>, N. Lewis<sup>a</sup>, Y. Bornat<sup>a</sup>, A. Daouzli<sup>a</sup>, M. Rudolph<sup>b</sup>, A. Destexhe<sup>b</sup>, S. Saïghi<sup>a</sup>

<sup>a</sup> IMS, University of Bordeaux, ENSEIRB, CNRS UMR5218, 351 cours de la Libération, F-33405 Talence Cedex, France

<sup>b</sup> UNIC, CNRS UPR2191, 1 avenue de la Terrasse, F-91198 Gif-sur-Yvette, France

### ARTICLE INFO

#### Article history:

Received 13 February 2008

Received in revised form 29 January 2010

Accepted 19 February 2010

#### Keywords:

Spiking neural networks

Integrated circuits

Hardware simulation

Conductance-based neuron models

Spike-Timing-Dependent Plasticity

### ABSTRACT

Many hardware-based solutions now exist for the simulation of bio-like neural networks. Less conventional than software-based systems, these types of simulators generally combine digital and analog forms of computation. In this paper we present a mixed hardware–software platform, specifically designed for the simulation of spiking neural networks, using conductance-based models of neurons and synaptic connections with dynamic adaptation rules (Spike-Timing-Dependent Plasticity). The neurons and networks are configurable, and are computed in ‘biological real time’ by which we mean that the difference between simulated time and simulation time is guaranteed lower than 50  $\mu$ s. After presenting the issues and context involved in the design and use of hardware-based spiking neural networks, we describe the analog neuromimetic integrated circuits which form the core of the platform. We then explain the organization and computation principles of the modules within the platform, and present experimental results which validate the system. Designed as a tool for computational neuroscience, the platform is exploited in collaborative research projects together with neurobiology and computer science partners.

© 2010 Elsevier Ltd. All rights reserved.

### 1. Introduction

Computational neuroscience commonly relies on software-based processing tools. However, there are also various hardware-based solutions which can be used to emulate neural networks. Some of these are dedicated to the simulation of Spiking Neural Networks (SNN), and take into account the timing of input signals by precisely computing the neurons’ asynchronous spikes. Neuron models can precisely describe the biophysics of spikes (action potentials) by computing the currents flowing through cell membrane and synaptic nodes. It is possible to reduce the size of these models to facilitate their computation. Other popular models are based on a phenomenological description of the neurons. They are well adapted to the study of complex network dynamics in neural coding or memory processing. While software tools can be configured for different types of models (Brette et al., 2007; Hines & Carneval, 1997) hardware-based SNN are dedicated to a given type of model. They may even be completely specialized, i.e. compute only one specific SNN model. In such a case, the hardware is designed for a specific application, as for example in the case of bio-medical artefacts (Akay, 2007).

Our group has been designing and exploiting neuromimetic silicon neurons for ten years (Renaud, Laflaquière, Bal & Le Masson,

1999, LeMasson, Renaud, Debay, & Bal, 2002, Renaud, Le Masson, Alvado, Saïghi, & Tomas, 2004, Renaud, Tomas, Bornat, Daouzli, & Saïghi, 2007). We have developed specific integrated circuits (IC) from biophysical models following the Hodgkin–Huxley (HH) formalism, in order to address two fields of research: (i) build a hardware simulation system for computational neuroscience to investigate plasticity and learning phenomena in spiking neural networks; (ii) develop the hybrid technique, which connects silicon and biological neurons in real time. The system presented in this paper belongs to the first category, although it may be quite simply adapted for hybrid configurations. This platform was specifically designed for the simulation in biological real time of SNN using conductance-based models of neurons and synaptic connections: it enables the construction of bio-realistic networks, and offers the possibility of dynamically tuning the model parameters. The models are derived from the Hodgkin–Huxley formalism (Hodgkin & Huxley, 1952), and rely strongly on the physiological characteristics of neurons. The platform can run simulations of small networks of point neurons modeled with up to 5 conductances, using different cortical neuron model cards. Kinetic synapse models have been implemented to simulate the network connections. Each of the latter can be characterized by its own adaptation function, following a programmable rule for Spike-Timing-Dependent Plasticity (STDP). The SNN is computed in biological real time (the system ensures that the difference between simulation times and simulated time is less than 50  $\mu$ s).

\* Corresponding author.

E-mail address: [sylvie.renaud@ims-bordeaux.fr](mailto:sylvie.renaud@ims-bordeaux.fr) (S. Renaud).

In Section 2, we describe the various models classically implemented on hardware simulators, and review various simulation platforms. We discuss the relationship between the SNN formats and the architecture of the platforms, and discuss our technical solution. In Section 3 we provide a detailed description of the architecture of the simulation platform (PAX), and present its specifications and key features. In Section 4 we present experimental results obtained using PAX: benchmark simulations to validate the implemented hardware, and open experiments to study biologically relevant SNN. Finally, we provide our conclusions and present the future versions of the PAX platform: technical evolutions of the various elements, and experiments to be implemented in a collaborative project between physicists and biologists.

## 2. Hardware-based SNN platforms

There are various ways in which SNN models can be computed, ranging from software to hardware implementations. Dedicated software tools are well known (Brette et al., 2007) and widely distributed. Although offering numerous models and parameters, they often have the drawback of requiring prohibitively long computation times when it comes to simulating large and complex neural networks. Recent improvements have been achieved using parallel and distributed computation (Johansson & Lansner, 2007; Migliore, Cannia, Lytton, Markram, & Hines, 2006). These systems use a large computation power to process complex networks rather than guarantee real-time performance. Another approach is to build dedicated hardware to process a network of predefined neuron types. Hardware approaches, like the one we describe in the present paper, have very interesting properties in terms of computation time but a higher development cost and constraints on the computed models. Therefore these systems are generally application-dedicated. Applications can range from purely artificial experiments, in particular the investigation of adaptation and plasticity phenomena in networks, to experiments on hybrid biological/artificial networks.

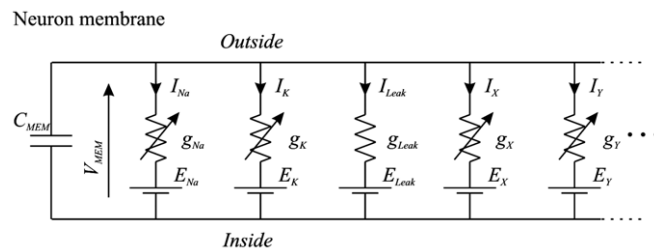
For such systems, designers generally consider simplified neuron models and balance imprecision by a higher number of neurons in the network (Fieres, Schemmel, & Meier, 2006; Indiveri, Chicca, & Douglas, 2006). An other approach is to reduce the number of the parameters in the system (Farquhar & Hasler, 2005), or to model neurons populations (Fieres et al., 2006; Indiveri et al., 2006; Renaud et al., 2004). Considering the network structure, some systems limit synaptic connectivity, others guarantee 'all-to-all' connected networks with uniform connection delays, but with less neurons. Finally, systems differ by their properties in terms of computation time, as some systems aim to simulate SNN "as fast as possible" (Indiveri et al., 2006), and other guarantee a fixed simulation timescale (Bornat, Tomas, Saïghi, & Renaud, 2005; Fieres et al., 2006).

In the following section we provide a summary of the principal models used for spiking neurons and synaptic connections. We describe several types of hardware-based SNNs and compare it to ours; this review highlights the diversity of possible solutions, and their respective relevance to various applications.

### 2.1. The model choice issue

#### 2.1.1. The neuron level

Most SNN models are point neuron models, in which a neuron represents a single computational element, as opposed to compartmental models which take into account the cells' morphology. Different levels of abstraction are possible, to describe point neuron models. One can focus on the properties of each ionic channel, or prefer to choose a behavioral description. The user needs to select the model by finding an acceptable compromise between two



**Fig. 1.** Equivalent electrical circuit for HH-based neuron models. In this example, additional conductances ( $x, y, \dots$ ) modulate the influence of the Na, K and leak channels.

contradictory criteria: faithfully reproduce the membrane voltage dynamics ( $V_{MEM}$ ), and minimize the computational load on the simulation system.

In the most detailed family of models, known as conductance-based models, ionic and synaptic currents charge and discharge a capacitor representing the neuron membrane (Gerstner & Kistler, 2002). All of these models find their origins in the Hodgkin and Huxley (1952) model (HH). Each ionic channel (Sodium: Na, Potassium: K...) is represented by a time- and voltage-dependent conductance: this electrophysiological description makes these models particularly well suited to an implementation involving analog electronics. Hodgkin–Huxley derived models have the same structure and include a larger number of types of ionic channel, in order to fit the fast and slow dynamics of the neurons. This multiplicity of models enables the diversity of biological neurons to be suitably represented (see Fig. 1). The main advantage of this formalism is that it relies on biophysically realistic parameters and describes individual ionic and synaptic conductances for each neuron in accordance with the dynamics of ionic channels. This type of model is necessary to emulate the dynamics of individual neurons within a network.

Conductance-based models reduced to 2 dimensions are also very popular, as they can be entirely characterized using phase plane analysis. The well-known FitzHugh–Nagumo (FN) (FitzHugh, 1961) and Morris–Lecar models (Morris & Lecar, 1981) are also worthy of mention.

Threshold-type models are another class of widely used models for SNN. They describe, at a phenomenological level, the threshold effect in the initiation of an action potential. The shape of the  $V_{MEM}$  signal is not reproduced by such models, and ionic currents are no longer processed. These models are adjusted by fitting the timing of the spikes and setting the threshold level. In terms of computational cost, they are interesting for the study of neural coding and the dynamics of large networks. The Integrate-and-Fire model (IF) and Spike Response Model (SRM) belong to this family (Gerstner & Kistler, 2002). The Leaky Integrate-and-Fire model (LIF), which although simple is able to compute the timing of spikes, is widely used for hardware SNN implementations (see Fig. 2). The LIF model computes  $V_{MEM}$  on a capacitor, in parallel with a leak resistor and an input current. When  $V_{MEM}$  reaches a threshold voltage, the neuron fires and its dynamics are neutralized during an absolute refractory period. Many models were derived from the LIF, and take into account (for example) the effects of modulation currents (Brette & Gerstner, 2005; Gerstner & Kistler, 2002; Izhikevich, 2004).

#### 2.1.2. The network level

The dynamics of a SNN and the formation of its connectivity are governed by synaptic plasticity. Plasticity rules formulate the modifications which occur in the synaptic transmission efficacy, driven by correlations in the firing activity of pre- and post-synaptic neurons. At the network level, spikes are generally

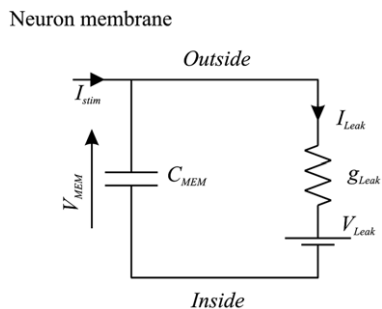


Fig. 2. Equivalent electrical circuit for the LIF neuron model.

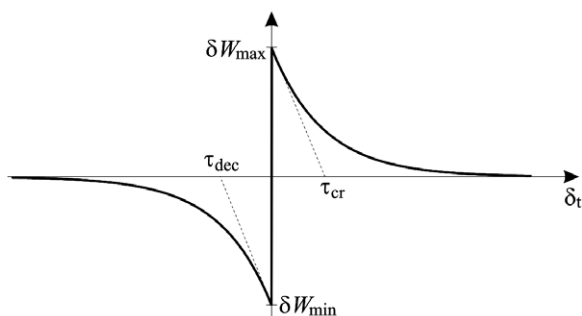


Fig. 3. Variation of synaptic weight  $\delta W_{ji}$  from neuron  $j$  (pre-synaptic) to neuron  $i$  (post-synaptic), as a function of the time interval  $\delta t = t_i - t_j$  ( $t_i$ : post-synaptic spike time,  $t_j$ : pre-synaptic spike time).

processed as events, and the synaptic weight  $W_{ji}$  (connection from neuron  $j$  to neuron  $i$ ) varies over time, according to learning rules. Among learning rules, Spike-Timing-Dependent Plasticity (STDP) algorithms (Gerstner & Kistler, 2002) are often used in hardware-based SNN. Fig. 3 illustrates the principle of standard STDP: when a post-synaptic spike arises after a pre-synaptic spike ( $\delta t > 0$ ), the connection is reinforced ( $\delta W_{ji} > 0$ ), whereas in the opposite case it is weakened. STDP requires dynamic control of the synaptic connectivity (Badoual et al., 2006), and may lead to significant computational costs if applicable to all synapses. For large scale networks, the connectivity between neurons becomes a critical issue and all-to-all adaptive connectivity becomes the to-be-achieved computational goal.

## 2.2. A review of hardware-based SNN

This section illustrates the diversity of SNN platforms. It will be seen that different applicative fields can lead to different implementations, and that the computational share between analog hardware, digital hardware and software can be varied. We compare the different SNN platforms in terms of model accuracy, network size and processing speed.

In the last 15 years, few hardware-based SNN systems were developed: in Fig. 4 we list the most representative or most recently developed of these (Binczak, Jacquir, Bilbault, Kazantsev, & Nekorkin, 2006; Glackin, McGinnity, Maguire, Wu, & Belatreche, 2005; Graas, Brown, & Lee, 2004; Hasler, Kozoil, Farquhar, & Basu, 2007; Indiveri & Fusi, 2007; Liu & Douglas, 2004; Renaud et al., 2007; Schemmel, Brüderle, Meier, & Ostendorf, 2007; Sorensen, DeWeerth, Cymbalyuk, & Calabrese, 2004; Vogelstein, Malik, & Cauwenberghs, 2004), as well as pioneer platforms (Jung, Brauer, & Abbas, 2001; LeMasson et al., 2002; Mahowald & Douglas, 1991). The systems make use of multi-compartmental (Hasler et al., 2007) and point neuron (Binczak et al., 2006; LeMasson et al., 2002; Mahowald & Douglas, 1991; Renaud et al., 2007; Sorensen et al., 2004) conductance-based models, or threshold-type models (Glackin et al., 2005; Indiveri & Fusi, 2007; Jung et al.,

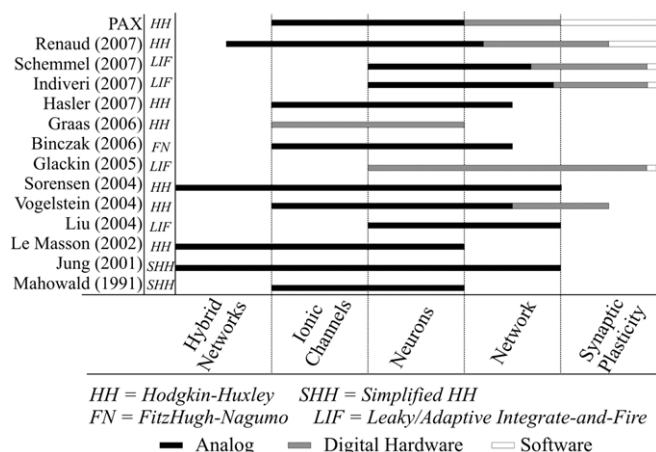


Fig. 4. Analog/digital and hardware/software distribution in state-of-the-art SNN systems; Horizontally: simulation features; detailed descriptions are given by: (Mahowald & Douglas, 1991) for Mahowald (1991); (Jung et al., 2001) for Jung (2001); (LeMasson et al., 2002) for le Masson (2002); (Liu & Douglas, 2004) for Liu (2004); (Vogelstein et al., 2004) for Vogelstein (2004); (Sorensen et al., 2004) for Sorensen (2004); (Glackin et al., 2005) for Glackin (2005); (Binczak et al., 2006) for Binczak (2006); (Hasler et al., 2007) for Hasler (2007); (Indiveri & Fusi, 2007) for Indiveri (2007); (Schemmel et al., 2007) for Schemmel (2007); (Renaud et al., 2007) for Renaud (2007).

2001; Liu & Douglas, 2004; Schemmel et al., 2007; Vogelstein et al., 2004) of SNN. The neurons are organized into networks of various sizes. Some platforms can be used for hybrid networks experiments (Jung et al., 2001; LeMasson et al., 2002; Sorensen et al., 2004). Fig. 4 illustrates the modeling complexity and processing equipment needed by these systems.

All of the SNN presented here are partially or entirely implemented onto hardware. Only 2 platforms are fully digital hardware systems (Glackin et al., 2005; Graas et al., 2004), using FPGAs (Field Programmable Gate Array) circuits for large scale LIF SNNs or for small scale HH SNNs.

All other solutions rely on analog computation using specifically designed integrated circuits (Application Specific Integrated Circuits – ASICs) at the neuron level, and even for plasticity. The simulated neural element can either be integrated onto a single chip or distributed over multiple chips, depending on the integration constraints. Analog integrated circuits are used to implement the previously described models (LIF, FN, HH). We have labeled SHH the HH inspired models, where some of the conductance functions are simplified or fitted, as opposed to the complete Hodgkin–Huxley description of non-linear, voltage- and time-dependent ionic conductances. In the case of an analog mode implementation, the signals are available as continuous variables, both in time and value. As a consequence, the simulation timescale can be precisely determined, for example real time or accelerated. In our case, real-time (electrical time = biological time) and biologically relevant neuron models make it possible to construct mixed living artificial networks, in which the silicon neurons are interconnected with the biological cells to form “hybrid networks”.

Digital design is characterized by much lower manufacturing costs, and has the advantages of an improved time-to-market performance, and straightforward re-configurability. However, analog SNN has two distinct advantages: a higher integration density, since a single wire encodes one signal (instead of  $N$  wires needed to represent a digital signal with  $N$ -bits), and a higher computational power, which is obtained by exploiting the intrinsic computational features of electronic elements such as transistors.

For large scale SNN, it may be useful to accelerate the simulation by one or more orders of magnitude (Schemmel et al., 2007). For such applications, LIF neuron models are generally implemented, using simple analog cells to simulate a point neuron.

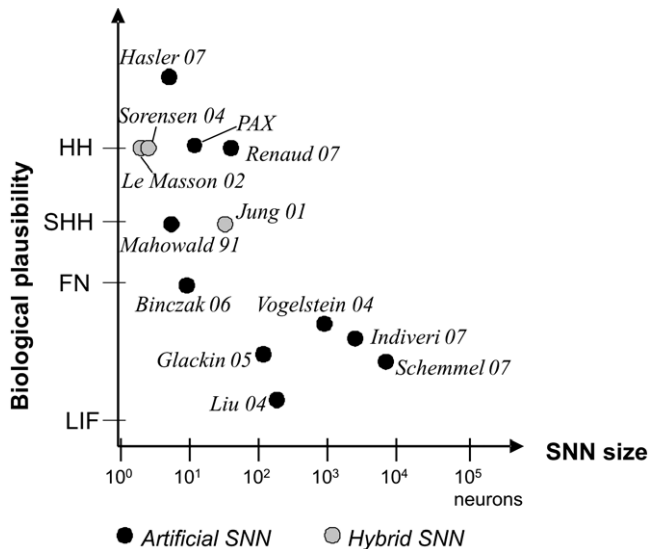


Fig. 5. Comparison of SNN hardware platforms, expressed in terms of model complexity versus network size.

However, when the system reaches the network level or integrates plasticity rules, digital hardware is preferred for connectivity computation and/or control. Event-based protocols are used to interface the neurons' activity (spike events) to the connectivity calculation element. An increasing number of neuromimetic systems present such mixed Analog/Digital (A/D) architectures, as shown in Fig. 4. In the end, the architecture of a given SNN platform is the result of a compromise between computational cost and model complexity, the latter of which also constrains the achievable network size. Fig. 5 illustrates this compromise and shows the current limits of such systems. To position the PAX system among hardware-based neural simulators, we can summarize its main features: conductance-based neurons models, small networks with all-to-all connectivity, flexible implementation of STDP on all synapses, real-time simulations.

### 3. The PAX simulation system

Our research group has been developing neuromorphic ASICs for more than 10 years. These ASICs have been used to form the core of various simulation platforms, designed in collaboration with neurophysiologists to emulate neural networks in biologically relevant configurations (Bornat et al., 2005; LeMasson et al., 2002; Renaud, Laflaquière, Bal, & Le Masson, 1999; Renaud et al., 2007). We describe here the PAX platform (for Plasticity Algorithm Computation System), which brings up additional features such as configurable networks and STDP. PAX is dedicated to the simulation of cortical neural networks, with a focus on the dynamics at the single neuron level. It was developed in the context of a collaborative project (EU project FACETS, FP6-IST-FETPI-2004-15879) whose aim is to lay the theoretical and experimental foundations for the realization of novel computing hardware, and which exploits the concepts experimentally observed in the brain. Preliminary experiments using the system under development were described in Zou et al. (2006).

Within the FACETS project, two hardware platforms have been developed to emulate SNN. This paper is related to the first one, called PAX. It is dedicated to the study of small cortical neural networks, with emphasis on the dependence of its dynamics on synaptic plasticity rules, mainly STDP. The second FACETS platform is developed in the University of Heidelberg (Schemmel et al., 2007) and integrates, at wafer scale, large networks of LIF-modeled neurons in accelerated time. Using optimal STDP rules validated by the PAX platform, this system should be able to emulate substantial fractions of the brain.

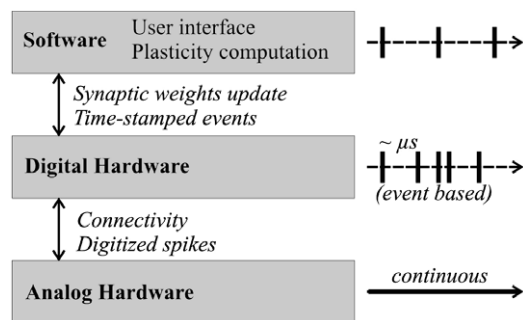


Fig. 6. PAX platform processing layers and data flow. The timing of computations is specific to each layer.

#### 3.1. General architecture

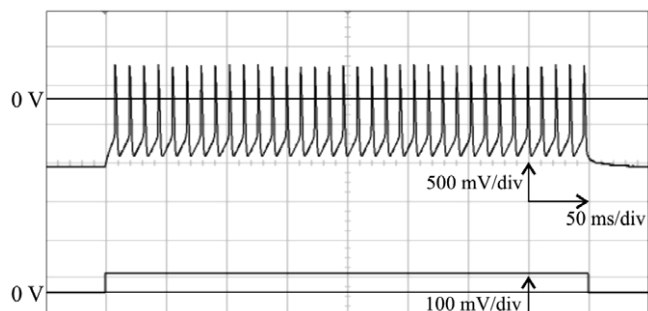
In the PAX system, an active neuron presents action potentials – also called “spikes” – at low frequencies (from 0.1 to 300 Hz). Spikes can be spontaneous, or be induced by synaptic stimulations from other neurons. The exact timing of spikes is a key piece of information on the neural network's activity and the evolution of its connectivity. The synaptic interactions can be subject to short-term and long-term variations, and may process cognitive mechanisms that rely on plasticity and learning. The PAX system computes the neurons' activity ( $V_{MEM}$ ) in real time using analog hardware. Synaptic interactions, which control the transmission of information between neurons, are processed digitally.

The PAX simulation system is organized into 3 layers (Fig. 6). The analog hardware layer runs continuous computations of the neurons' activity on analog ASICs, and the configuration of the latter is controlled by the digital hardware layer. This hardware is also in charge of collecting spike event information from the analog neurons, and controls the synaptic connectivity, which feeds back to the analog hardware. In order to optimize computational speed, the processing mode is event-based, with a 1 μs timing resolution. Predefined stimulation patterns (such as the emulation of background cortical activity (Destexhe, Rudolph, Fellous, & Sejnowski, 2001)) can also be applied to individual neurons. The next layer includes the software driver and interface, which are in charge of controlling the bi-directional data transfer to the software via a PCI bus. Finally, a PC running a real-time operating system hosts software functions used to compute the plasticity and to update the synaptic weights of the neural network. The software layer also includes user interface functions to control the off-line and on-line simulation configurations. Hardware requests to the software layer are event-based. It significantly increases the available computational power and thus allows the implementation of power consuming plasticity algorithms.

#### 3.2. The analog hardware layer

Computational models of spiking neurons are based on experimental data related to the electrophysiological behavior of neurons in different cortical areas (Connors & Gutnick, 1990; Gupta, Wang, & Markram, 2000; McCormick & Bal, 1997).

Our goal is to represent the “prototypical” types of neurons and synaptic interactions present in neocortex. To this aim, we therefore need to capture the main intrinsic firing and response properties of excitatory and inhibitory cortical neurons. We have used conductance-based spiking neuron models to reproduce the two main neuronal types in neocortex, according to the classification of Connors and Gutnick (1990). Simple Hodgkin and Huxley (1952) type models describe precisely the firing characteristics of the cells. All neurons contain a leak conductance and voltage-dependent Na and K conductances, which are essential for the generation of action potentials. This model reproduces



**Fig. 7.** Inhibitory (FS) neuron activity computed and measured on the analog hardware (“Trieste” ASIC). The lower trace is the input voltage, controlling  $I_{STIM}$ , applied to the ASIC. The upper trace shows variations of the neuron membrane voltage in response to the stimulation current  $I_{STIM}$ .

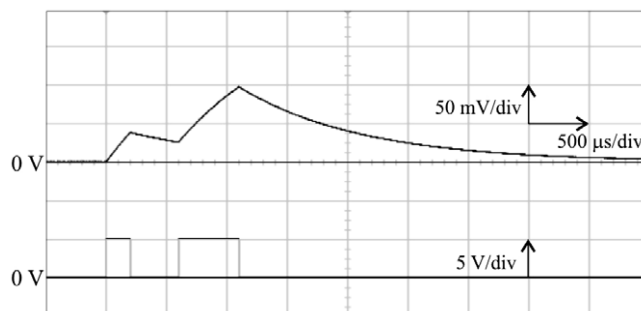


**Fig. 8.** Excitatory (RS) neuron activity computed and measured on the analog hardware (“Trieste” ASIC). The lower trace is the input voltage, controlling  $I_{STIM}$ , applied to the ASIC. The upper trace shows variations of the neuron membrane voltage in response to the stimulation current  $I_{STIM}$ .

the “fast spiking” (FS) phenotype (Fig. 7), which represents the major class of inhibitory neurons in different cortical areas, as well as in many other parts of the central nervous system. The model for excitatory neurons includes an additional slow voltage-dependent potassium current ( $I_{MOD}$ ) responsible for spike-frequency adaptation. It corresponds to the “regular spiking” (RS) phenotype (Fig. 8), which represents the major class of excitatory neurons in the cerebral cortex. In the first generation of ASIC used in the PAX systems, i.e. the “Trieste” circuit, a one-bit input ( $E/\bar{I}$ ) configures the hardware neuron to be excitatory when  $E = 1$ , or to be inhibitory when  $E = 0$ . When configured to emulate RS neurons ( $E = 1$ ), the circuit computes one among four possible values for the potassium conductance  $\bar{g}_{MOD}$  using 2 dedicated digital inputs. This value can be changed on-line during the simulation while the other parameters of the RS and FS models are fixed. To ease simulations and eventually compensate the ASIC offset currents, a voltage-controlled current source is implemented in each “neural element”, to be used a stimulation current source  $I_{STIM}$ .

The equations used in the models were adapted to match the constraints of hardware implementation. The time constants of the activation and inactivation variables are voltage-dependent in the Hodgkin–Huxley original model. At ASIC level, we simplified the equations by setting the time constants for activation of the sodium and potassium currents to a constant value. For the inactivation of the sodium current and the activation of the modulation current ( $I_{MOD}$ ), we implemented two-level time constants. These simplifications have only minor consequences on the model’s behavior (Zou et al., 2006), as they essentially change the shape of the spikes. The equations and parameters computed for the model conductances are given in Appendix A.

To increase dynamic range and noise immunity, we apply  $\times 10$  gain factor (electronic signals =  $10 \times$  biological signals) for both voltages and conductances. That choice led to a  $\times 1.00$  scale factor



**Fig. 9.** Synaptic current computed and measured on the analog hardware (“Trieste” ASIC). The upper trace shows the exponentially decaying synaptic current (measured as a voltage across a  $100 \text{ k}\Omega$  resistance), triggered by successive binary pulses; the pulse width (lower trace) encodes the synaptic weight.

for time, a  $\times 100$  scale factor for currents and  $\times 10$  scale factor for capacitance values. Although the 1.00 simulation timescale is constraining the design, it paves the way to direct interfacing with living tissues without risking any information loss. Despite the simplifications on time constants, this conductance-based model is among the more accurate models ever implemented in hardware.

The models used to represent synaptic interactions are pulse-based kinetic models of synaptic conductances (Destexhe, Mainen, & Sejnowski, 1994) (see Appendix B for the equations), which describe the time course and summation dynamics of the two principal types of synaptic interactions in the central nervous system (respectively AMPA and GABA\_A post-synaptic receptors for excitation and inhibition). The dynamics of the synaptic conductances are captured using a two-state (open-closed) scheme, in which opening is driven by a pulse. This model can easily handle phenomena such as summation or saturation, and accurately describes the time course of synaptic interactions (Fig. 9). Furthermore, it is rather easily integrated into hardware in the form of multi-synapse elements. Pre-synaptic events are collected at a single input to generate the pre-synaptic pulse. The width of this pulse, which triggers the transition to the opening state, can be modulated. Together with  $\bar{g}$  (maximum conductance, see Appendix B), it encodes the strength of the multi-synapse, and is dynamically updated (Bornat et al., 2005). The resulting synaptic strength, controlled by the digital layer, cover 3 orders of magnitude, which is particularly interesting to simulate complex networks dynamics.

Full-custom ASICs were designed and fabricated using BiCMOS  $0.8 \mu\text{m}$  technology (Fig. 10), based on a full-custom library of electronics function (Bornat et al., 2005), validated by previous designs. Each mathematical function present in the neuron model corresponds to an analog module in the library, and receives tunable analog inputs corresponding to some of the model parameters ( $I_{STIM}$ ,  $\bar{g}_{MOD}$ ,  $\bar{g}$  for synapses). The modules are arranged to generate electronic currents, which emulate ionic or synaptic currents. Finally, the computational cores of the ASICs are “neural elements”, each of which is able to process the membrane voltage of 1 cortical neuron (inhibitory or excitatory). It also includes a threshold comparator used for spike detection and two synaptic conductances, following respectively the AMPA-type (excitatory) and GABA\_A-type (inhibitory) kinetic models. Considering that each spike detector converts the membrane potential  $V_{MEM}$  to a 1-bit code, each neural element outputs two representations of  $V_{MEM}$ : the continuous analog value and the asynchronous digitized spikes.

### 3.3. The digital hardware layer

The digitized outputs of each “neural element” are directly connected to FPGA inputs. Once captured by the FPGA, these

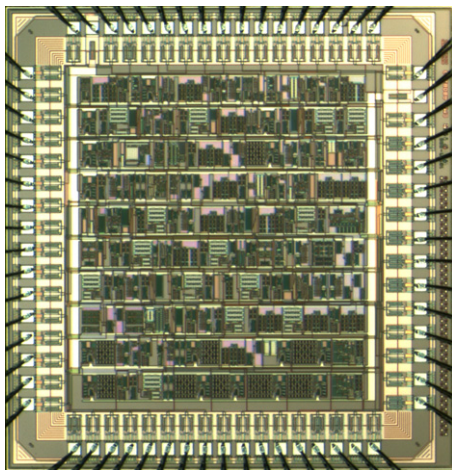


Fig. 10. Microphotograph of a “Trieste” ASIC (area 10 mm<sup>2</sup>). The layout of this device is full-custom. The ASIC was manufactured by AMS (austriamicrosystems).

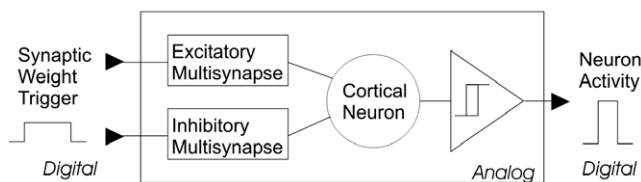


Fig. 11. Structure and I/O for a hardware neural element.

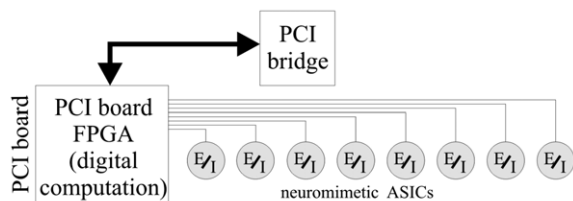


Fig. 12. Internal architecture of the full-custom PCI board. The ASICs can be configured to simulate excitatory (*E*) or inhibitory (*I*) neurons.

signals represent spike ‘events’, which are computed by the upper layers of the system. The digital hardware layer is also in charge of generating the synaptic weight-triggering signal (Fig. 11). Using the multi-synapse scheme, the neural network can handle all-to-all connections, whatever the number of neurons in the network.

Both digital and analog layers are physically assembled onto a full-custom printed circuit board (Figs. 12 and 13). Eight configurable “Trieste” ASICs, associated with discrete components, make up the analog layer. The digital layer is comprised mainly of FPGAs (SPARTAN2E400<sup>TM</sup> from Xilinx Inc. clocked at 64 MHz) and a PCI9056<sup>TM</sup> component supplied by PLX Technology, which constitutes the PCI Bridge. This board can be plugged into any PC-compatible computer with a PCI slot.

The simulation of a neural network is based on three major tasks:

(1) Ensuring network connectivity: this should be performed each time an ASIC generates an action potential. Such events are time-stamped, i.e. the timing of the event is evaluated by the FPGA. Two processing flows are generated inside this FPGA. The first of these sends available synaptic stimulations to the ASICs multi-synapse inputs, according to a predefined connectivity matrix stored on the digital layer. This matrix contains the weights of all possible connections in the simulated network (a non-existing connection is coded as a connection with a null weight). Once created, the synaptic stimulations are sent to the post-synaptic neuron. The

second processing flow time stamps the events and stores the data (neuron number and timing of the spike) into a FIFO. Due to the structure of the digital layer, the time required for a single connectivity operation is in all cases less than 1  $\mu$ s.

(2) Computing the plasticity: the software computes the STDP algorithms to determine the evolution of synaptic weights for each plastic connection, and to update the weights in the connectivity matrix accordingly (see plasticity loop in Fig. 14). Weights are stored as 8-bit integers and sent to the analog layer. The 8-bit values code the length of the stimulus sent to the synapse. The relationship between this integer and the stimulation length is configurable and can be updated during simulation.

(3) Sending external patterns of synaptic stimulation, which correspond to brain background activity and stimuli: the stimulation pattern is stored in the computer’s memory; sub-patterns are buffered on the board memory, with on-the-fly updates carried out by the software during the simulation. Although this approach is complex to handle and has a certain computational cost, the computer’s high storage capacity can be used for very long stimulation patterns for which the board’s local memory would be insufficient. It is also more convenient to use the dynamic memory allocation available on the computer, because the allocated space can be adapted to the complexity of the stimulation pattern.

In the PAX system, the analog ASICs communicate with a dedicated digital environment hosted by a PC-compatible computer, using a PCI exchange bus. The components and configuration of the computer were chosen to reduce overall system latency and PCI bus usage (high speed transfers, USB and other unused controllers disabled at the BIOS level).

### 3.4. The software layer

The host computer runs on a real-time patched GNU/Linux<sup>®</sup> operating system based on the Ubuntu<sup>®</sup> distribution. With an open source system, we can choose precisely which elements of the system need to be activated for our application, and which ones can be disabled. Many unused features have been effectively disabled in order to minimize the PC processor tasks. The X graphical server is disabled, but monitoring is still possible in console mode or through a remote graphical connection using the LAN (Local Area Network) interface.

All simulations are preceded by a configuration phase involving the definition and transmission of a set of parameters (neural network topology), and the preparation of external stimulation patterns (if needed). During the effective neural network simulation, the software runs as a server daemon, i.e. it enters into a supposedly infinite loop. Due to the limited number of neurons (maximum 8) available for the current version of PAX, we have defined an event-based process with a maximum latency of 10  $\mu$ s. During each loop, the following operations are sequentially performed:

- (1) Read the stimulation data from the FIFO on the board.
- (2) Compute the plasticity algorithms and the resulting changes in synaptic weights according to the STDP algorithms (see Appendix C).
- (3) Update the connectivity matrices stored both in the PC memory and in the hardware system.

A library of interface functions, optimized for fast and easy use of the platform, has been written. As an example, sleeping states are handled by the reading function. With this approach, the user can ask for reading data without needing to know whether any are available. When an event is available, it is transmitted. If the data buffer is empty, the reading function enters a sleep state; it quits that state when it receives a hardware interrupt, at the arrival of a new event. We also took particular care to avoid time-demanding instructions within the source code.

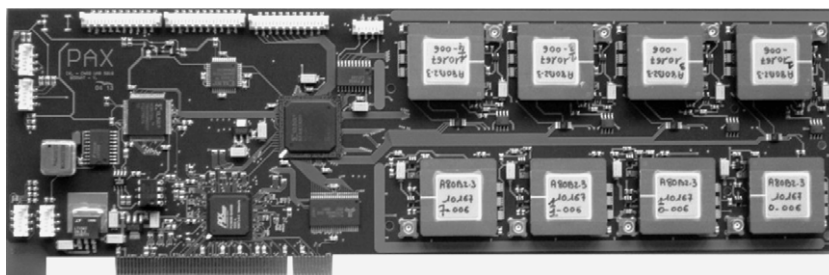


Fig. 13. Photograph of the PAX PCI board that supports; from left to right, PCI Bridge, FPGA and 8 ASICs.

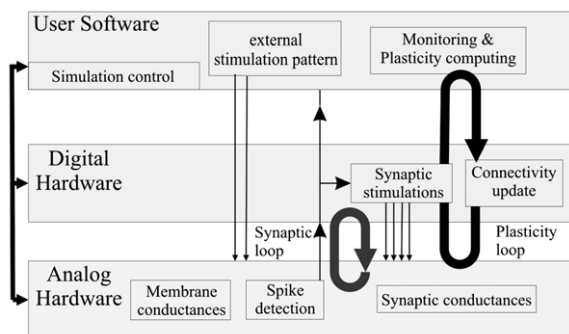


Fig. 14. Distribution of SNN computational tasks over the different layers of the PAX system.

The library contains functions, which handle the definition and use of stimulation patterns. Looping features are available, to use patterns which correspond to periods shorter than that of the complete simulation. The user can fix different pattern lengths for each neuron, and thereby avoid parasitic correlation effects: although the pattern on a single synapse can be repetitive, the global network stimulation pattern is non-periodic.

The software layer optimizes the flexibility and computation power of the system. It computes exact equations of different plasticity models, which is necessary to exploit at best the accuracy of the Hodgkin and Huxley model used for the neurons. The exploration of neural networks dynamics at different time scales (from single event to learning sequence) is a specificity of this hardware simulation system.

### 3.5. Evolution of the PAX platform

With the PAX platform reported here, we were able to demonstrate the feasibility of a conductance-based simulation system mixing analog and digital hardware, together with computations run on software. By running various experiments on PAX, we were able to identify the limitations that should be overcome on future platforms. The neuron models are computed in continuous time on the analog hardware layer, whereas connectivity is computed on the digital layer and plasticity is computed by the software layer, with a maximum latency of 10  $\mu$ s on each event. When plasticity is implemented on a desktop computer, the bottleneck to the increase of the networks size is computational power. Depending on the complexity of the plasticity algorithm and the nature of the network, only networks with a small number of neurons ( $N$ ) can be simulated (between 10 and 30). To increase  $N$ , the STDP calculations can be transferred to the digital hardware layer so that the system becomes completely event-based: the stimulations and synaptic weight updates are all performed by an FPGA and the software layer is used only to provide a user interface dedicated to the network configuration and the display/storage of neural state information. In such a configuration, the digital hardware layer has to perform the following tasks (Fig. 15):

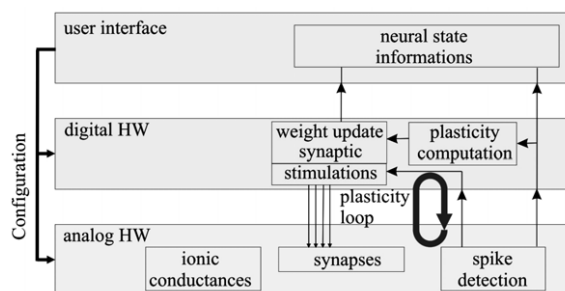
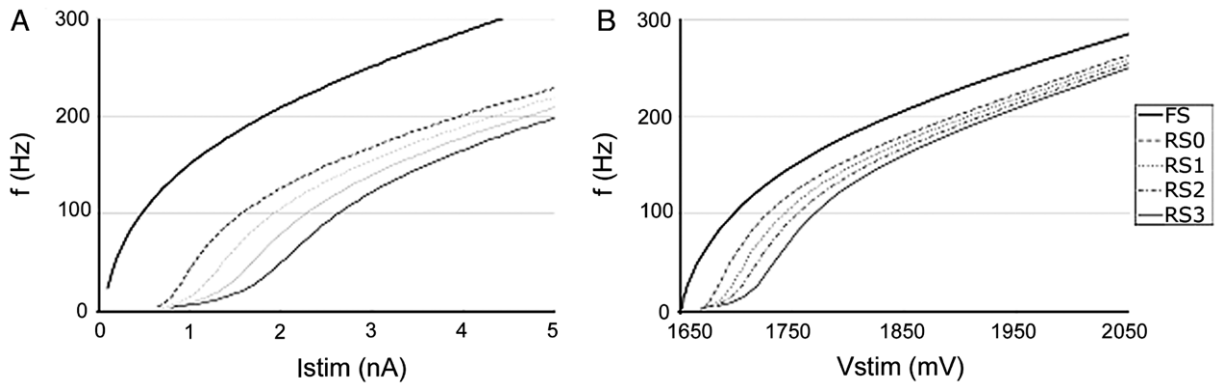


Fig. 15. Distribution of SNN computational tasks in the follow-up version of the PAX system.

- Receive the neurons' configuration parameters from the user via the software layer, and send them to the corresponding ASICs;
- Map the topology of the neural network (connectivity and plasticity);
- Send the neural state information (time-stamped spike events, synaptic weights evolution) back to the user;
- Receive spike events; send the synaptic input signal to post-synaptic neurons;
- Compute the STDP algorithms in real time and update the synaptic weights.

We have tested different architectures capable of performing such tasks on a Spartan3™ (XC3S1500FG456). The limiting factor is the number of multipliers available on the FPGA: one multiplier is needed for each single  $W_{ji}$  calculation, and these must be pipelined to perform the necessary multiplications of the STDP (4 for the STDP in Appendix C). In a fully parallel architecture, if all of the 132 available multipliers (32 original ones and 100 implemented in LUT) are used, the network will be limited to the simulation of 132 single plastic connections, as opposed to a total of  $N^2$  possibilities. In a fully sequential architecture with a sampling period of 1.5  $\mu$ s and a maximum latency time of 150  $\mu$ s, the FPGA would thus be able to compute 625 connections, i.e. 25 all-to-all connected neurons with a spike occurring every sampling period. This worst case is of course not biologically realistic, but points out the limitations of such architecture.

The authors are now in the process of designing the next version of PAX, which will be based on the above solution, i.e. entirely hardware-based on-line computation. The technical choices (FPGA size, number of ASICs controlled by a FPGA) are currently being optimized, to enable conductance-based networks of 100–200 neurons to be simulated in real time with the same accuracy, with an average connectivity and spiking frequency. On this new system, computation limits are solved by hardware, and the bottleneck to increase the network size becomes real-time low-latency communication.



**Fig. 16.** Frequency versus stimulation current curves, for software and hardware models. The differences between software and hardware are due to mismatch in the fabrication process. (A) Software simulations. (B) Hardware simulations, where  $V_{STIM}$  is the voltage applied to the ASIC. With this generation of ASIC, we have no external access to ISTIM.

#### 4. Experimental results

One application of the PAX system is to study the influence of specific parameters (type of neurons, STDP algorithms) on the plasticity efficiency of a cortical neural network. To obtain results with good biological relevance, the system first needs to be benchmarked by studying different configurations that validate all of the system's features. In this section, we present experimental results obtained with the PAX system. These results successively validate the implementation of conductance-based neurons on ASICs, connectivity simulations on the FPGA, and the computation of background activity together with plasticity.

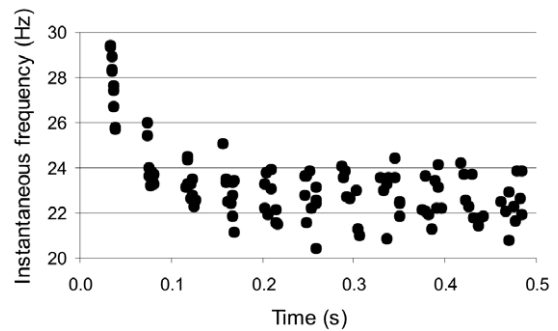
##### 4.1. Single neuron simulation

In this experiment, two ASICs were configured (by appropriate setting of the model parameters in the internal RAM) to simulate the different models described in 3.1.1: the 3-conductance FS model, and the 4-conductance RS model in its 4 configurations (4 different values for  $\bar{g}_{MOD}$ , as described in Appendix A). We measured each neuron spiking frequency as a function of the stimulation current, and plotted the  $f(I)$  curves shown in Fig. 16(B). These measurements were compared with simulations of the same models using Neuron software (Fig. 16(A)).

Although the plots shown in Fig. 16 demonstrate that stable spiking frequencies are achieved, this is not sufficient to describe the adaptation phenomenon, which exists in RS neurons, as it is a temporal phenomenon. When a stimulation current is applied to a neuron, it begins to oscillate at a high instantaneous frequency, which then decreases until it reaches a stabilized frequency. The adaptation process results from the slow  $I_{MOD}$  current, which emulates the dynamics of the calcium and calcium-dependent potassium channels. In order to observe the dynamics of adaptation processes we plotted, for an RS neuron with  $\bar{g}_{MOD,3} = 136.8 \mu\text{S}/\text{cm}^2$ , the instantaneous spiking frequency for 10 simulation trials (Fig. 17). The resulting adaptation is dependent on  $\bar{g}_{MOD}$  and  $\tau_{m,MOD}$  (see equations in Appendix A). This figure illustrates one of the consequences of analog processing: due to electronic noise at transistor level, the simulations are not strictly reproducible (which is not the case with digital computation). This phenomenon is taken advantage of in our experiments: it is helpful when observing the effects of cortical noise and neuronal diversity in a neural network.

##### 4.2. Simulation of an adaptive network of neurons

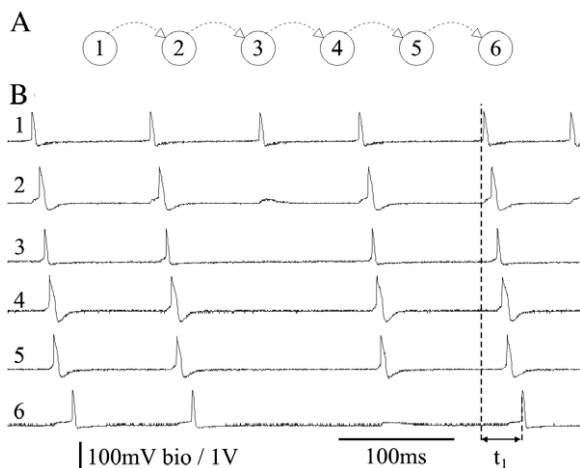
In this section we present the simulation of an adaptive network implemented on the PAX system. We chose to model adaptation



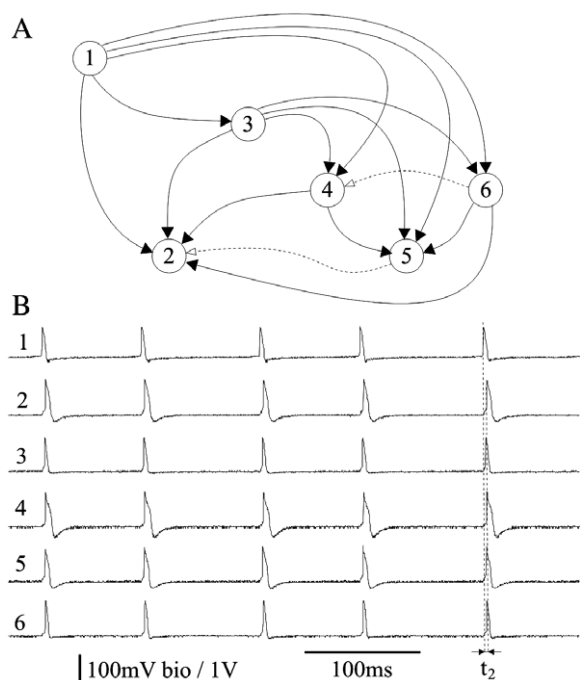
**Fig. 17.** Instantaneous frequency measurements (10 trials) on RS3 neurons. Stimulation occurs at time  $t = 0$ .

effects by using the STDP model described in Appendix C, which includes LTP, LTD, soft bounds and eligibility criteria. The network we present comprises 6 RS neurons, each of which is connected to all of the others by an excitatory and adaptive synaptic connection. The neurons' model parameters and stimulation currents were tuned to set them to a potential just under the spiking threshold. When stimulated, their spiking frequencies are slightly different. Neurons 1 and 3 are the fastest, followed by neurons 4 and 6, then neurons 2 and 5. In its initial state (Fig. 18), 5 synaptic weights were tuned to weak, but non-zero values. The network therefore forms a simple transmission chain. At the beginning of the simulation, neuron 1 was stimulated to fire at a low frequency (about 10 Hz), and this activity then propagates along the chain. Note that due to analog computational noise, the propagation time  $t_1$  of the whole chain can vary between 10 ms and 40 ms in successive experiments. Sooner or later, one neuron fails to fire and propagation ceases.

Starting from the initial state described in Fig. 18, the simulation system computes network activity and connectivity in real time. It converges to a final pattern, which happens to be constrained by the neurons' intrinsic firing rate. Fig. 19 shows the resulting final network connectivity and activity. This final state is obtained after a delay ranging between 10 and 40 s. The activity of the neurons is globally synchronized within the network, since they all fire within a time window of less than 4 ms. However, differences still exist between the neurons: if it is considered that a spike on neuron 1 triggers the network activity, neuron 6 will fire simultaneously with neuron 4, but before neurons 2 and 5. In terms of network architecture, if neuron 1 is considered to represent the input layer and neuron 6 the output layer, neurons 2, 4 and 5 become useless. Neuron 3, due to its fast spiking property, improves the transmission delay between the input and the output. The STDP has



**Fig. 18.** Beginning of the simulation. (A) Network connectivity: all cells are RS neurons with four conductances. Five synaptic weights have small initial values (digital values with a maximum of 200:  $W_{12} = 190$ ;  $W_{23} = 140$ ;  $W_{34} = 155$ ;  $W_{45} = 120$ ;  $W_{56} = 200$ ). All other connections exist, but have a null weight. (B) Activity pattern: membrane potentials of neurons 1 to 6. The voltage scale is indicated for the biological model (100 mV) and as measured on the hardware neurons (1 V). Neuron 1 spikes autonomously, whereas the activity of the other neurons is triggered by their synaptic inputs in accordance with the above scheme.



**Fig. 19.** End of the simulation: the network reaches a stable state. (A) Network connectivity: full lines indicate connections with high synaptic weights; dotted lines indicate connections with low synaptic weights. Connections with null weights are not indicated (digital values where maximum is 200:  $W_{12} = 199$ ;  $W_{13} = 199$ ;  $W_{14} = 198$ ;  $W_{15} = 197$ ;  $W_{16} = 199$ ;  $W_{32} = 199$ ;  $W_{34} = 197$ ;  $W_{35} = 199$ ;  $W_{36} = 197$ ;  $W_{42} = 199$ ;  $W_{45} = 199$ ;  $W_{52} = 117$ ;  $W_{62} = 199$ ;  $W_{64} = 130$ ;  $W_{65} = 199$ ). (B) Activity pattern: membrane potentials of neurons 1 to 6. The voltage scale is indicated for the biological model (100 mV), and as measured on the hardware neurons (1 V). All neurons spike within a time window ( $t_2$ ) of less than 4 ms.

“optimized” the network functionality as a transmission chain by classifying the neurons and reinforcing connections between the faster ones. It should be noted that the network is also more robust, because different paths do exist between the input (neuron 1) and the output (neuron 6): the accidental misfiring of a neuron will be by-passed, and will not stop the transmission.

### 4.3. Simulation of an adaptive neural network with synaptic noise inputs

#### Experiments

We have studied the activity of an excitatory neuronal network with all-to-all connectivity and a STDP algorithm, together with the influence of synaptic noise inputs with different rates and correlations. In this experiment, all features of the PAX system were exploited: as described in Section 3, stimulation patterns can be stored for each neuron and applied as an external input during the simulation, while respecting the real-time processing requirement. For each neuron, we created uncorrelated stimulation patterns of Poisson-trains inputs with an average rate of 10 Hz. These patterns emulate a cortical random synaptic activity (Destexhe et al., 2001). The simulated network is the 6-neurons network with STDP described in Appendix C, in which each neuron receives the Poisson stimulation pattern. The experimental protocol is then as follows: Phase A: all synaptic weights are initially set to zero and the neurons are set to a resting potential just below their spiking threshold. For 360 s (biological time and simulation time are identical), we recorded spike trains from all neurons as well as the evolution of the synaptic weights. Phase B: the initial synaptic weights were not set to zero. We ran one trial with all weights equal to a maximum value (soft bound), and three trials with randomized initial values. The outputs were recorded again for a period of 360 s. Phase C: we increased the correlation between the input noise sources. With different correlation rates, we used different initial weights, and ran the simulation again for 360 s.

#### Stimulation input patterns with different correlation rates

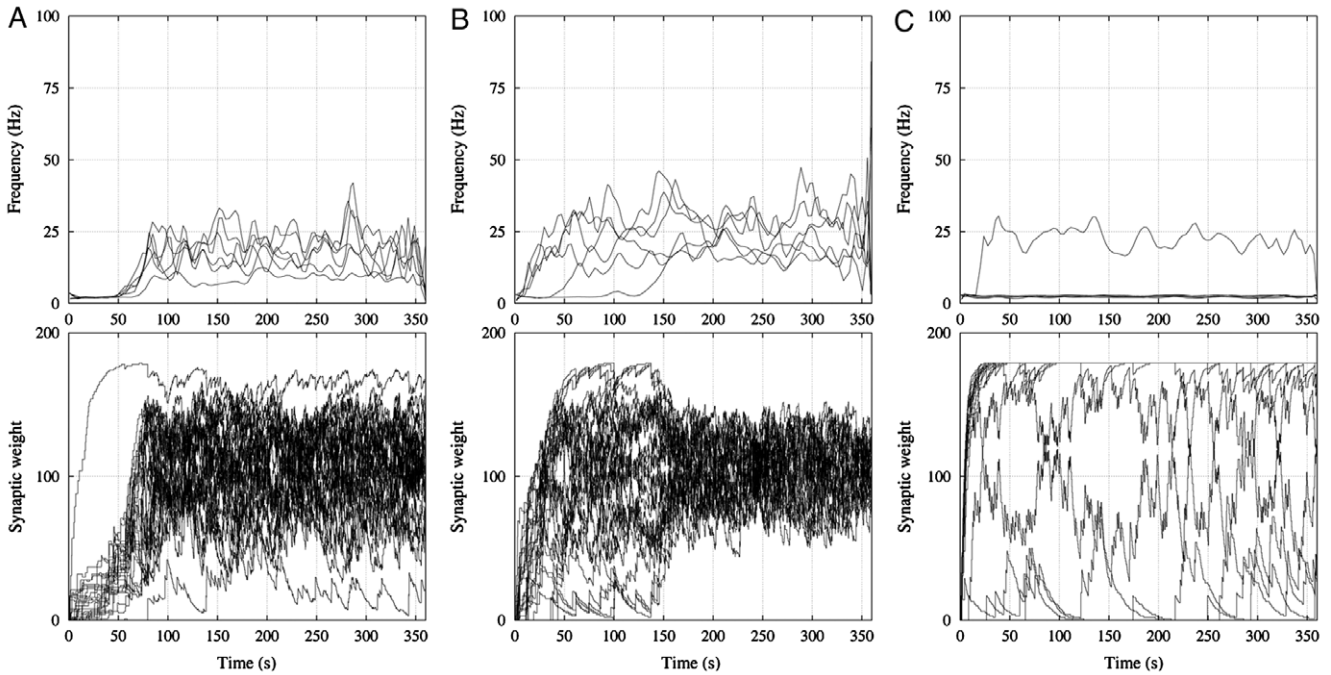
In order to define patterns, which emulate synaptic noise, we compute the time intervals between synaptic inputs. These intervals are constrained by a Poisson distribution, and their mean value corresponds to the synaptic noise period, noted here as  $m$ . In order to generate  $n$  correlated noise sources, as required in Phase C of the experimental protocol, we generated a single Poisson distribution of time intervals,  $X$ . This distribution is obtained as follows:  $X = \frac{\{x_1, \dots, x_n\}}{x_i} = N(0, 1) \cdot \sqrt{m - \frac{1}{2}} + m$  where  $N(0, 1)$  is a normal distribution and  $m$  is the average interval.  $X$  is converted into an absolute time pattern  $Y : X = \{x_1, \dots, x_n\} \rightarrow Y = \{y_1, \dots, y_n\}$ , then  $Y : y_i = \sum_{j=1}^i x_j$ . The  $n$  noise input patterns are generated with one event around the  $Y$  event. The time interval between the event  $Y$  and the pattern event is given by  $\varepsilon : \varepsilon = N(0, 1) \cdot (1 - \alpha) \cdot \frac{m}{6}$  where  $N(0, 1)$  is a normal distribution,  $m$  is the average interval and  $\alpha \in [0, 1]$  is the correlation coefficient.

#### STDP parameters

In accordance with the STDP Algorithms given in Appendix C, the parameters equations are based on a precise biophysical model. The parameters for the exponential constants are  $W_{LTP} = 0.1$ , describing potentiation, and  $W_{LTD} = 0.005$  for depression. The time constants are  $\tau_p = 14.8$  ms for the potentiation exponential of  $P$ , and  $\tau_Q = 33.8$  ms for the depression exponential of  $Q$ . The eligibility  $\varepsilon$  (influence of previous spikes from the same neuron) has an exponential time constant for the pre-synaptic neuron ( $\tau_{\varepsilon_j} = 28$  ms), and for the post-synaptic neuron ( $\tau_{\varepsilon_i} = 88$  ms) (Froemke & Dan, 2002). This takes into account features such as frequency dependence and spike triplets. Details of the STDP algorithmic implementation and parameters are provided in Bornat et al. (2005).

#### Results

To illustrate the influence of synaptic noise correlation on STDP efficiency, we present in Fig. 20 the results for initially null synaptic weights, input synaptic noise with a mean frequency of 10 Hz



**Fig. 20.** Simulations of a 6 RS neurons network with all-to-all excitatory synapses with STDP. The mean frequency of input synaptic noise is 10 Hz, initial weights are null. Upper plots: instantaneous frequency of the 6 neurons versus time. Lower plots: synaptic weights of the 36 synapses versus time. (A) Uncorrelated input synaptic noise,  $\alpha = 0$ . (B) Correlation of input synaptic noise,  $\alpha = 0.8$ . (C) Full correlation of input synaptic noise,  $\alpha = 1$  (same input pattern on each neuron).

( $m = 100$  ms), and different values of the correlation rate  $\alpha$ . Fig. 20 shows the evolution of the synaptic weights to their steady state: a rather large distribution (Fig. 20(A)) is obtained for small  $\alpha$  values; a limited range distribution (Fig. 20(B)) for intermediate  $\alpha$  values; a quasi-bipolar distribution (Fig. 20(C)) for  $\alpha = 1$ . In the latter case, the synaptic weights achieve only the maximum or the minimum value. The network “discriminates” the input noise correlation.

By producing many sets of initial conditions (initial weights and correlation rates), and by repeating each experiment 10 times, we carried out more than one hundred simulations of 360 s each. For such a case, the PAX system, with its unconditionally real-time processing capability, offers a very clear advantage when compared to software solutions.

## 5. Conclusion

In this paper we have presented a platform for the simulation of small conductance-based neural networks with adaptation. The rationale for such a development is to provide a tool for computational neuroscience, which can guarantee real-time processing. The PAX platform relies on analog hardware to compute neuronal activity, and digital hardware or software to compute the network’s dynamic connectivity. Mixed analog–digital systems are an emerging solution for the emulation of spiking neural networks, as shown in Section 2, in which various models and implementation techniques for hardware-based platforms have been reviewed. As in many other fields of microelectronics (RF embedded systems for example), a mixed implementation offers the advantages of both solutions: analog circuits have a higher integration density, and digital platforms have better programmability. The PAX system will evolve whilst maintaining its already mixed architecture (see Section 3). A newly designed analog ASIC will replace the “Trieste” ASIC currently used in the system. Based on the same library of functions as that used in the previous device, it computes ionic conductances with entirely programmable model parameters. The parameters are stored on-chip in DRAM memory cells, and can be modified by the user at any time. As explained in

Section 3.5, STDP functions will be computed using FPGA hardware in the majority of cases. Non-standard STDP functions, or some STDP algorithms in highly connected networks, will be computed by the software. In any case, the computation will be distributed in order to optimize data processing and ensure that real-time performance is maintained. Finally, the system will have the capacity to address biological diversity, in terms of neuron types as well as plasticity rules.

Real-time performance is an important feature for two main reasons: (1) it guaranties the computational speed, compared to traditional software processing tools (although massive parallel computing facilities do provide high performance, this is achieved at tremendous cost – which strictly limits the extent to which they can be used); (2) it provides a simplified interactivity with the natural environment, such as sensory inputs. The PAX platform is designed to be an efficient and convenient tool for the exploration of biologically realistic networks. As shown in Section 4, simulations, which explore the behavior of a neural network in multi-parameter space, are time-costly. The PAX system helps us to study the relatively complex paradigms involving plasticity mechanisms.

## Appendix A. Equations of the models, and biological values of the parameters used in the conductance-based models implemented on the ASICs

Membrane voltage (electrical circuit in Fig. 1):

$$C \frac{dV_{MEM}}{dt} = -I_{Na} - I_K - I_{Leak} - I_{MOD} + I_{STIM}$$

with  $C_{MEM} = 1 \mu\text{F}/\text{cm}^2$  and the area of an integrated cell =  $0.00022 \text{ cm}^2$ .

Ionic current:

$$I_{Ion} = \bar{g}_{Ion} \cdot x^p \cdot y^q (V_{MEM} - E_{Ion})$$

with

$$\begin{cases} \tau_x \cdot \frac{dx}{dt} = x_\infty - x \\ x_\infty = \frac{1}{1 + \exp\left(-\frac{V_{MEM} - V_{OFFSET,x}}{V_{SLOPE,x}}\right)} \end{cases}$$

and

$$\begin{cases} \tau_y \cdot \frac{dy}{dt} = y_\infty - y \\ y_\infty = \frac{1}{1 + \exp\left(+\frac{V_{MEM} - V_{OFFSET\_y}}{V_{SLOPE\_y}}\right)} \end{cases}$$

Sodium current:

$$I_{Na} = \bar{g}_{Na} \cdot m^3 \cdot h (V_{MEM} - E_{Na})$$

with

$$\bar{g}_{Na} = 0.05 \text{ S/cm}^2; \quad E_{Na} = 50 \text{ mV}; \quad \tau_m = 0.03 \text{ ms};$$

$$V_{OFFSET\_m} = -37 \text{ mV}; \quad V_{SLOPE\_m} = 7.2 \text{ mV};$$

$$\tau_h = \begin{cases} 3 \text{ ms} & \text{if } V_{MEM} > 0 \\ 0.25 \text{ ms} & \text{if } V_{MEM} < 0; \end{cases} \quad V_{OFFSET\_h} = -42 \text{ mV}$$

$$\text{and } V_{SLOPE\_h} = -4.6 \text{ mV}.$$

Potassium current:

$$I_K = \bar{g}_K \cdot n^4 (V_{MEM} - E_K)$$

with

$$\bar{g}_K = \begin{cases} 0.01 \text{ S/cm}^2 & \text{if inhibitory (FS) neuron} \\ 0.005 \text{ S/cm}^2 & \text{if excitatory (RS) neuron;} \end{cases}$$

$$E_K = -100 \text{ mV}; \quad \tau_n = 3 \text{ ms}; \quad V_{OFFSET\_n} = -37 \text{ mV} \quad \text{and}$$

$$V_{SLOPE\_n} = 11.38 \text{ mV}.$$

Leak current:

$$I_{Leak} = \bar{g}_{Leak} (V_{MEM} - E_{Leak})$$

with

$$\bar{g}_{Leak} = \begin{cases} 0.001 \text{ S/cm}^2 & \text{if inhibitory (FS) neuron} \\ 0.0015 \text{ S/cm}^2 & \text{if excitatory (RS) neuron} \end{cases}$$

and

$$E_{Leak} = \begin{cases} -70 \text{ mV} & \text{if inhibitory (FS) neuron} \\ -80 \text{ mV} & \text{if excitatory (RS) neuron.} \end{cases}$$

Modulator current generator:

$$I_{MOD} = \begin{cases} 0 & \text{if inhibitory (FS) neuron} \\ \bar{g}_{MOD} \cdot m (V_{MEM} - E_{MOD}) & \text{if excitatory (RS) neuron} \end{cases}$$

with

$$E_{MOD} = -100 \text{ mV}; \quad \tau_m = \begin{cases} 300 \text{ ms} & \text{if } V_{MEM} > 0 \\ 8 \text{ ms} & \text{if } V_{MEM} < 0; \end{cases}$$

$$V_{OFFSET\_n} = -35 \text{ mV}; \quad V_{SLOPE\_n} = 11.4 \text{ mV},$$

depending on the user specification

$$\begin{cases} \bar{g}_{MOD\_1} = 45.5 \mu\text{S/cm}^2 \\ \bar{g}_{MOD\_2} = 90.9 \mu\text{S/cm}^2 \\ \bar{g}_{MOD\_3} = 136.8 \mu\text{S/cm}^2 \\ \bar{g}_{MOD\_4} = 181.8 \mu\text{S/cm}^2. \end{cases}$$

## Appendix B. Models for synaptic currents

We chose to model synaptic interactions using “exponential” synapses, where the synaptic conductance increases by a given “quantal conductance” when a pre-synaptic spike occurs, then relaxes exponentially to zero. The associated post-synaptic current  $I_{SYN}$  is given by:

$$I_{SYN} = \bar{g} \cdot r (V_{MEM} - E_{SYN}), \quad \text{and} \quad \frac{dr}{dt} = \alpha [T] \cdot (1 - r) - \beta \cdot r$$

where  $\bar{g}$  is the maximum conductance,  $E_{SYN}$  the reverse synaptic potential,  $V_{MEM}$  the post-synaptic membrane potential,  $r$  the fraction of receptors in open state,  $\alpha$  and  $\beta$  the voltage-independent forward and backward rate constants, and  $[T]$  the transmitter concentration.

As the quantum  $\Delta g$  is proportional to the pulse width  $\Delta t$ , the latter parameter is exploited to modulate  $\Delta g$ . Furthermore, synaptic summation, which occurs when multiple pre-synaptic spikes occur simultaneously, will be handled naturally by the integration of successive transmitter pulses. The time constants and quantum sizes can be adjusted to fit experimental recordings of excitatory and inhibitory synaptic currents. The translation between digital signals and the conductance quantum is achieved via a pulse duration control system. As long as the signal is active at the input, the synaptic conductance increases. When the end of the “pulse” is reached, the conductance decreases exponentially. The choice of this model is partly justified by the ease with which it can be implemented onto hardware: it enables multiple synaptic inputs to be concatenated on a unique mechanism. The strength of the synapse is tunable, via the duration of the pulse signals, which are controlled by a digital interface.

We have chosen the following values, to fit realistic synaptic models:

$$\text{excitatory synapse: } \alpha = 1.1e^6 \text{ M}^{-1} \text{ s}^{-1} \quad \beta = 190 \text{ s}^{-1}$$

$$E_{SYN} = 0 \text{ V}$$

$$\text{inhibitory synapse: } \alpha = 5e^6 \text{ M}^{-1} \text{ s}^{-1} \quad \beta = 180 \text{ s}^{-1}$$

$$E_{SYN} = -80 \text{ mV}.$$

The time constants of the synapses are calculated using  $R = 33 \text{ k}\Omega$ ,  $C = 68 \text{ nF}$  for the excitatory synapse and  $R = 100 \text{ k}\Omega$ ,  $C = 100 \text{ nF}$  for the inhibitory one.

When a spike occurs in the pre-synaptic neuron, a transmitter pulse is triggered such that  $[T] = 1 \text{ mM}$ , for a period of 1 ms. The synaptic conductance of excitatory synapses, from RS to FS, is  $\bar{g}_{exc} = 0.01 \text{ mS}$ , and for inhibitory synapses, from FS to RS, is  $\bar{g}_{inh} = 0.005 \text{ mS}$ .

## Appendix C. Synaptic plasticity models

The plasticity algorithms are based on Spike-Timing-Dependent Plasticity (STDP), which is believed to be the most significant phenomenon involved in the adaptation in neural circuits. This plasticity depends on the relative timing of pre-synaptic and post-synaptic spikes. Using a single connection from  $N_j$  to  $N_i$ , the simplest STDP algorithm describes 2 phenomena: a long-term potentiation (LTP), noted  $P(t)$ , and a long-term depression (LTD), noted  $Q(t)$ , according to:

$$\begin{aligned} \frac{dW_{ji}}{dt} = & \sum_{t_i} P(t - t_j^{last}(t)) d(t - t_i) \\ & - \sum_{t_j} P(t - t_i^{last}(t)) d(t - t_j) \end{aligned}$$

where  $t_j^{last}(t)$  and  $t_i^{last}(t)$  give the last spike occurrences at time  $t$ , in neurons  $N_j$  and  $N_i$ .

Potentiation means that the weight  $W_{ji}$  increases when the post-synaptic neuron  $N_i$  spikes at time  $t_i$  after a spike of the pre-synaptic neuron  $N_j$  at time  $t_j$ . Depression means that the weight  $W_{ji}$  decreases when the pre-synaptic neuron  $N_j$  spikes at time  $t_j$ , after a spike of the post-synaptic neuron  $N_i$  at time  $t_i$ . LTP and LTD are illustrated in Fig. 21.

The functions are estimated from experimental results on real neural networks.  $P(t)$  and  $Q(t)$  are exponential functions:

$$P(t) = \exp(-t/\tau_p) \quad \text{and} \quad Q(t) = \exp(-t/\tau_Q).$$

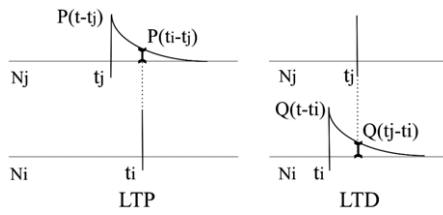


Fig. 21. Long-term potentiation and long-term depression functions, depending on the occurrence of events on  $N_i$  and  $N_j$ .

Using this model, the relative change of synaptic weight is shown in Fig. 3, where  $(t_i - t_j)$  represents the time difference between an event on the post-synaptic neuron  $N_i$  and the pre-synaptic neuron  $N_j$ .

We can take two more STDP effects into account: the memory effect (Badoual et al., 2006), and the saturation effect. The resulting expression is:

$$\frac{dW_{ji}}{dt} = e_i e_j \left\{ (W_{LTP} - W_{ji}) \sum_{t_i} P(t - t_j^{last}(t)) d(t - t_i) - (W_{LTD} - W_{ji}) \sum_{t_j} P(t - t_i^{last}(t)) d(t - t_j) \right\}.$$

The term  $\varepsilon$ , called spike eligibility, indicates that the change of synaptic weight  $W_{ji}$  is less significant for a second occurrence of a pre-synaptic spike at time  $t_j^{last}(t)$ , than for the first occurrence at time  $t_j^{last-1}(t)$ . This memory effect can be modeled for  $N_j$  and  $N_i$  by:

$$\left\{ \begin{array}{l} \varepsilon_j = 1 - \exp\left(-\frac{t_j^{last} - t_j^{last-1}}{\tau_{\varepsilon_j}}\right) \\ \varepsilon_i = 1 - \exp\left(-\frac{t_i^{last} - t_i^{last-1}}{\tau_{\varepsilon_i}}\right) \end{array} \right.$$

In the previous model, there was no saturation effect for  $W_{ji}$ , which in biological terms is not entirely realistic. In order to address this problem, a maximum and a minimum saturation limit can be added to the STDP algorithms. These are called “soft-bound” values, respectively  $W_{LTP}$  and  $W_{LTD}$ .  $W_{ji}$ , and will be limited by  $W_{LTD} < W_{ji} < W_{LTP}$ .

References

Akay, M. (2007). *Handbook of neural engineering – Part II: neuro-nanotechnology: artificial implants and neural prosthesis*. Wiley IEEE Press.

Badoual, M., Zou, Q., Davison, A., Rudolph, M., Bal, T., Frégnac, Y., & Destexhe, A. (2006). Biophysical and phenomenological models of multiple spike interactions in spike-timing dependent plasticity. *International Journal of Neural Systems*, 16, 79–97.

Binczak, S., Jacquir, S., Bilbault, J.-M., Kazantsev, V. B., & Nekorkin, V. I. (2006). Experimental study of electrical FitzHugh–Nagumo neurons with modified excitability. *Neural Networks*, 19, 684–693.

Bornat, Y., Tomas, J., Saighi, S., & Renaud, S. (2005). BiCMOS analog integrated circuits for embedded spiking neural networks. In *Proceedings of the XX conference on design of circuits and integrated systems*.

Brette, R., & Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94, 3637–3642.

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. *Journal of Computational Neuroscience*, 23, 349–398.

Connors, B. W., & Gutnick, M. J. (1990). Intrinsic firing patterns of diverse neocortical neurons. *Trends Neurosciences*, 13, 99–104.

Destexhe, A., Mainen, Z. F., & Sejnowski, T. J. (1994). An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Computation*, 6, 14–18.

Destexhe, A., Rudolph, M., Fellous, J.-M., & Sejnowski, T. J. (2001). Fluctuating synaptic conductances recreate in-vivo-like activity in neocortical neurons. *Neuroscience*, 107, 13–24.

Farquhar, E., & Hasler, P. (2005). A bio-physically inspired silicon neuron. *IEEE Transactions on Circuits and Systems I*, 52, 477–488.

Fieres, J., Schemmel, J., & Meier, K. (2006). Training convolutional networks of threshold neurons suited for low-power hardware implementation. In *Proceedings of the 2006 international joint conference on neural networks* (pp. 21–28). IEEE Press.

FitzHugh, R. (1961). Impulses and physiological states in models of nerve membrane. *Biophysic Journal*, 1, 445–466.

Froemke, R. C., & Dan, Y. (2002). Spike timing-dependent synaptic modification induced by natural spike trains. *Nature*, 416, 433–438.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models: single neurons, populations, plasticity*. Cambridge University Press.

Glackin, B., McGinnity, T. M., Maguire, L. P., Wu, Q. X., & Belatreche, A. (2005). A novel approach for the implementation of large scale spiking neural networks on FPGA hardware. In *Proceedings of IWANN 2005 computational intelligence and bioinspired systems* (pp. 552–563).

Graas, E., Brown, E., & Lee, R. (2004). An FPGA-based approach to high-speed simulation of conductance-based neuron models. *Neuroinformatics*, 04, 417–435.

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

Hasler, P., Kozoil, S., Farquhar, E., & Basu, A. (2007). Transistor channel dendrites implementing HMM classifiers. In *Proceedings of IEEE international symposium on circuits and systems* (pp. 3359–3362).

Hines, M. L., & Carneval, N. T. (1997). The NEURON simulation environment. *Neural Computation*, 9, 1179–1209.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. *Journal of Physiology (London)*, 117, 500–544.

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

Indiveri, G., & Fusi, S. (2007). Spike-based learning in VLSI networks of integrate-and-fire neurons. In *Proceedings of IEEE international symposium on circuits and systems* (pp. 3371–3374).

Izhikevich, E. (2004). Which model to use for cortical spiking neurons. *IEEE Transactions on Neural Networks*, 15, 1063–1070.

Johansson, C., & Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Networks*, 20(1), 48–61.

Jung, R., Brauer, E. J., & Abbas, J. J. (2001). Real-time interaction between a neuromorphic electronic circuit and the spinal cord. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9(3), 319–326.

LeMasson, G., Renaud, S., Debay, D., & Bal, T. (2002). Feedback inhibition controls spike transfer in hybrid thalamic circuits. *Nature*, 4178, 854–858.

Liu, S.-C., & Douglas, R. (2004). Temporal coding in a silicon network of integrate-and-fire neurons. *IEEE Transactions on Neural Networks*, 15, 1305–1314.

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

McCormick, D. A., & Bal, T. (1997). Sleep and arousal: thalamo cortical mechanism. *Annual Review of Neurosciences*, 20, 185–215.

Migliore, M., Cannia, C., Lytton, W. W., Markram, H., & Hines, M. L. (2006). Parallel network simulations with NEURON. *Journal of Computational Neuroscience*, 21, 119–129.

Morris, C., & Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. *Biophysic Journal*, 35, 193–213.

Renaud, S., Laflaquière, A., Bal, T., & Le Masson, G. (1999). Analog circuits for modeling biological neural networks: design and applications. *IEEE Transactions on Biomedical Engineering*, 46(6), 638–645.

Renaud, S., Le Masson, G., Alvado, L., Saighi, S., & Tomas, J. (2004). A neural simulation system based on biologically-realistic electronic neurons. *Information Sciences*, 161(1–2), 57–69.

Renaud, S., Tomas, J., Bornat, Y., Daouzli, A., & Saighi, S. (2007). Neuromimetic ICs with analog cores: an alternative for designing spiking neural networks. In *Proceedings of IEEE international symposium on circuits and systems* (pp. 3355–3358).

Schemmel, J., Brüderle, D., Meier, K., & Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated I&F neurons. In *Proceedings of IEEE international symposium on circuits and systems* (pp. 3367–3370).

Sorensen, M., DeWeerth, S., Cymbalyuk, G., & Calabrese, R. L. (2004). Using a hybrid neural system to reveal regulation of neuronal network activity by an intrinsic current. *Journal of Neurosciences*, 24, 5427–5438.

Vogelstein, R. J., Malik, U., & Cauwenberghs, G. (2004). Silicon spike-based synaptic array and address-event transceiver. In *Proceedings of ISCAS'04* (pp. 385–388).

Zou, Q., Bornat, Y., Saighi, S., Tomas, J., Renaud, S., & Destexhe, A. (2006). Analog–digital simulations of full conductance-based networks of spiking neurons. *Networks: Computation in Neural Systems*, 17(3), 211–233.