Design exploration methodology for memristor-based spiking neuromorphic architectures with the Xnet event-driven simulator

O. Bichler, D. Roclin, C. Gamrat, D. Querlioz

Design Exploration Methodology for Memristor-Based Spiking Neuromorphic Architectures with the Xnet Event-Driven Simulator

O. Bichler, D. Roclin and C. Gamrat
CEA, LIST
91191 Gif-sur-Yvette Cedex, France
Email: olivier.bichler@cea.fr
Telephone: (+33)1.69.08.28.31
Fax: (+33)1.69.08.83.95

D. Querlioz
Institut d'Electronique Fondamentale,
Univ. Paris-Sud, CNRS
91405, Orsay, France
Email: damien.querlioz@u-psud.fr

***Abstract*—We introduce an event-based methodology, and its accompanying simulator ("Xnet") for memristive nanodevice-based neuromorphic hardware, which aims to provide an intermediate modeling level, between low-level hardware description languages and high-level neural networks simulators used primarily in neurosciences. This simulator was used to establish several results on Spike-Timing-Dependent Plasticity (STDP) modeling and implementation with Resistive RAM (RRAM), Conductive Bridge RAM (CBRAM) and Phase-Change Memory (PCM) type of memristive nanodevices. We present several simulation case studies that illustrate the event-based simulation strategies that we implemented, including unsupervised features extraction and Monte Carlo simulations. A discussion comparing event-based and fixed time-step simulation is included as well, and gives some metrics to guide the choice between the two depending on the application to simulate.**

I. Introduction

FOLLOWING the advancements in computational neuroscience, spiking neuromorphic hardware has gained momentum over the last years [1]–[4]. This trend is reinforced with the latest proposals to use memristive nanodevices as synapses, which are particularly attractive to implement efficient timing-based learning rules like Spike-Timing-Dependent Plasticity (STDP) in dense crossbar arrays [5]–[9]. A major focus of Spiking Neural Network (SNN) hardware is to capture biological processes with a much higher realism than earlier Artificial Neural Networks (ANN), thus enabling richer interactions with neuroscience, large-scale hardware-accelerated neural simulations and real-time behaving systems [10]. Another emerging field of applications for SNN are hardware Intellectual Property (IP) cores, especially in embedded computers, where efficiency is a major focus. SNN could indeed complement or replace otherwise computationally heavy sensor processing, like audio or video patterns extraction, learning, recognition and tracking. To model such systems, hardware description languages such as VHDL [11], Verilog or SystemC [12] do not provide the appropriate level of abstraction for fast and efficient architectural exploration, which generally implies tuning the network topology, neural parameters or learning rules depending on the intended task. At the opposite, neural network simulators popular in the neuroscience community such as Neuron [13], Brian [14] or NEST [15] can provide a higher level of abstraction. However, they lack the integration of synaptic memristive device modeling, hardware constraints and any custom features required for the targeted application.

Additionally, aggressively scaled down analog sub-threshold CMOS neurons and memristive nano-devices are plagued by device-to-device variations and intrinsic device variability to a large extend [16], [17]. Tacking into account this variability at a system level is becoming increasingly critical as it is more and more challenging to conceal it at the device level. This is however typically hard and expensive to tackle with both hardware description languages and high-level NN simulators.

To provide an intermediate modeling level for neuromorphic hardware, we introduce several event-based simulation strategies that are implemented in our event-driven simulator ("Xnet"). It is aimed to be a simple and effective simulator, which is designed from the beginning to integrate variability and stochasticity in both synaptic and neural models. With semi-physical synaptic device models, event-based simulation can also be used to estimate the synaptic power consumption of the system and to give direct feedback for technological optimization.

In the first section of this paper, we briefly introduce the event-based model used for memristive devices based neuromorphic hardware. Several case studies are presented in section II, namely event-driven spike shape modeling, unsupervised features extraction with LIF neuron model equipped with STDP and lateral inhibition introduced earlier, phase-change memory detailed device and system modeling, Monte Carlo simulations and genetic optimization of parameters. Finally, event-based simulation is compared with the Brian fixed time-step one in section III, in order to illustrate the main differences between the two types of simulation and give some metrics to guide the choice between the two depending on the parameters of the problem.

II. Methodology

The programming and reading pulses applied on the nanodevices (or "synapses") constitute the base events in our spiking neuromorphic hardware. They are sent or received by the networks nodes – its "neurons". These neurons are

This work has been supported by EU-FET grant "SYnaptic MOlecular NEtworks for Bio-inspired Information Processing (SYMONE)" (318597).

the active elements in the network and are implemented with analog or digital CMOS. In Xnet, the neurons are modeled functionally for computational efficiency. To simulate the system, we propose a mixed and flexible event processing model. The use of the C++ programming language [18] fulfills the need for a systems level language that provides high-level abstractions and fast simulations.

A. Events Processing

The simple yet flexible event processing engine, which is used in Xnet, is presented in figure 1. It is organized around an event queue, implemented with a priority queue of the C++ standard library. This turned out to be the most efficient structure for this purpose. The events are sorted by their timestamp. The event on top of the queue is therefore the next event scheduled to take place. When an event is processed, it induces new events. Their timestamp is calculated using the voltages in the system, and the models for the neurons and the synapses. The new events are inserted and sorted in the event queue. Events that have been processed are removed from the queue. An event can be a neuron spiking, or a voltage in the system changing. Some events cause some memristive nanodevices to change their conductance, and thus the system to learn.

This event processing engine differs from some spiking neural networks simulators that do not compute the events' exact timings (like SpikeNET [19]), but just model the events' order. Knowing the exact timings is necessary for using nanodevices models.

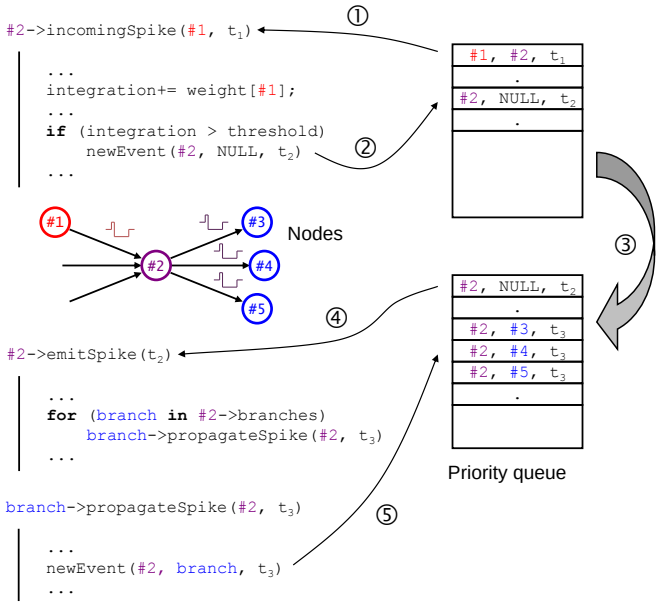

Fig. 1. Events processing in Xnet. The next scheduled event is processed by its destination node in the `incomingSpike()` method (1), which can create internal events (2) that are inserted into the priority queue (3). Internal events are processed by the `emitSpike()` method (4), which can then emit events towards output nodes by calling their `propagateSpike()` method (5).

B. Input Stimuli Generation

The input nodes of a network can be mapped to various types of external stimuli summarized in figure 2. Static frames and auditory stimuli can be temporally coded to emulate spiking retina- or cochlea-like sensors [20], [21]. Address Event Representation (AER) data can be loaded directly to form the input stimuli. AER is a standard asynchronous communication protocol, widely used in the neuromorphic hardware community for communication between spiking neurons.

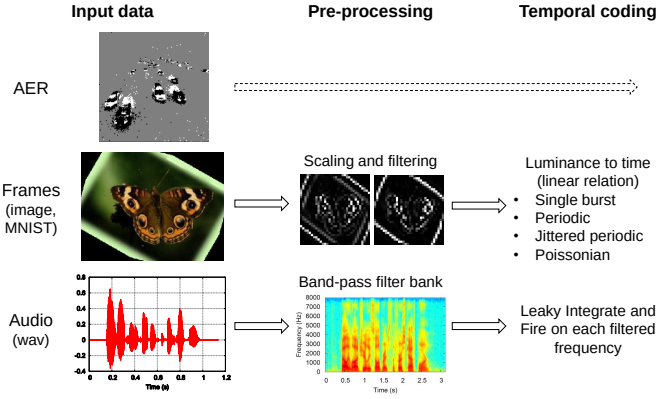

Fig. 2. Possible input stimuli: AER recording, static image and audio waveform. The pre-processing and temporal coding steps can be adapted to emulate hardware sensors or pre-processing units.

The basic building block to implement competitive learning is a group of neurons fully connected to the inputs with global lateral inhibition, as shown in figure 3. Lateral inhibition can be of any form, the simplest one implementing a winner-take-all between all the neurons. Considering its nano-scale hardware implementation, the canonical form of this building block is a crossbar of memristive synaptic devices.

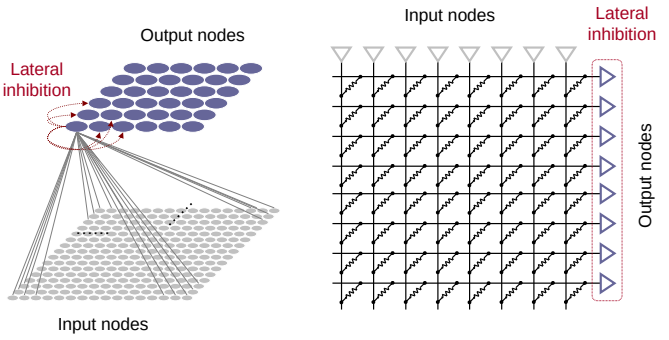

Fig. 3. Basic building block for a group of neurons, which can feature lateral inhibition. Left: topological view. Right: flattened view in a crossbar of synaptic elements.

C. Synapse and Neuron Models

Xnet is fundamentally synapse-centric. This differs from the neural network simulators which are neuron-centric, where the synapses are usually modeled as a single floating number parameter. In Xnet, the neurons and associated synaptic learning rule always emulate programming pulses on the synaptic

devices: the two rules used in our modeling methodology are (1) they cannot change the conductance of the nanodevices in absolute value, and (2) they do not have any knowledge of the current value of the conductance (no free read before write).

Functional and/or semi-physical modeling is possible with event-based modeling. We developed models for PCM, CBRAM and RRAM devices that fit experimental characterizations for selected programming conditions: programming pulses voltage, duration and initial device conductance state. For PCM, cumulative conductance increase through progressive crystallization with identical programming pulses is modeled. The models also include the variability and stochasticity in device conductance change. Each parameters is thus modeled with a normal or log-normal distribution, with a mean and a standard deviation: minimum/maximum conductance, conductance increment/decrement upon successive programming pulses, increment/decrement damping depending on the conductance of the device, or increment/decrement effective conductance change probability, which are the main parameters used for CBRAM and RRAM devices.

By choosing experimentally a fixed programming condition, in terms of pulse voltage and duration, a simple behavioral model for conductance change can be established, using the above-mentioned parameters, without requiring a deep understanding of the underlying physics.

III. Case Studies

In this section, several system-level simulation examples using Xnet are presented. They show the versatility of the simulator by integrating various hardware constraints such as voltage pulse width, lateral inhibition, phase-change memory behavioral modeling and system modeling. Monte Carlo simulations and genetic optimization of parameters capabilities are also presented.

A. Spike Shape Modeling

Although Xnet is an event-based simulator, modeling non-zero duration pulses is possible, without the performances penalty that would incur the switch to a fixed time steps simulation. Spike shape modeling is essential to model the more complex synaptic behaviors at the electrical level and reliably predict circuit performances at this level, while still working with a highly abstract description for the system and the network nodes, that can be either implemented with an analog or a digital circuit for example.

To allow spike shape modeling in Xnet, events have a type and each node in the network can define its own types. An example of a node modeling square pulses on its synapses is shown in figure 4. Event-based modeling is by nature only possible for piece-wise analytically solvable pulse and/or synaptic behavior. Moreover, because the simulation time is directly proportional to the number of events, multi-events spike modeling is suitable only up to a few events per spike or synaptic updates, compared to fixed time step simulations.

A square pulse model was used for the simulated results reported in [8], [22], which show unsupervised learning of the

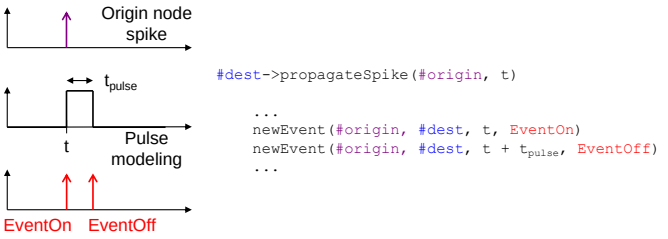

Fig. 4. Non-zero duration pulse modeling in Xnet. The `EventOn` and `EventOff` event types model respectively the rising edge and the falling edge of the electrical pulse.

MNIST handwritten digits database through STDP on RRAM memory devices with classification performances comparable to classical ANN. Homeostasis and semi-supervised learning were also shown, using a global reward mechanism, which is easy to implement at the network level in the simulator.

B. Unsupervised Features Extraction

In this example, we present a biologically inspired approach to extract temporally correlated patterns from an asynchronous event stream using unsupervised STDP learning implemented with memristive devices. We have simulated the development of selectivity in an array of 60 neurons receiving spikes from a 128×128 pixels silicon retina chip that generates spikes in response to local increases ("positive" events) and decreases ("negative" events) in luminance at the pixel level [23]. The neurons implement a special form of STDP, that increases the conductance of memristive devices that were activated just before a post-synaptic spike and decreases the conductance of all the other memristive devices uniformly. The memristives devices are modeled as in [24]. When tested with a recorded input sequence of 10 minutes of traffic on a freeway, the system develops neurons that respond to cars moving at particular locations within the image. The complete study, featuring a two-layers system and robustness to noise and variability analysis can be found in [25].

C. Phase-Change Memory System Modeling

The following case study present the implementation of the previous example using Phase-Change Memory (PCM) devices as synapses. PCM is a mature resistive memory technology, which is a good candidate for neuromorphic applications because of CMOS compatibility, high scalability, strong endurance, and good retention characteristics. The complete system was introduced in [26] and described in details in [27]. Because of the specific phase-change physical process, only the increase of PCM conductance can be made gradual with identical programming voltage pulses (shown in figure 6(a)), a scheme using two PCM devices per synapse was devised and validated with Xnet (figure 6(b)). For each connection, the synaptic state is stored in a special purpose `Synapse_PCM` object, inherited from the base `Synapse` object, containing two complete sets of parameters for the behavioral model exposed in figure 6(a).

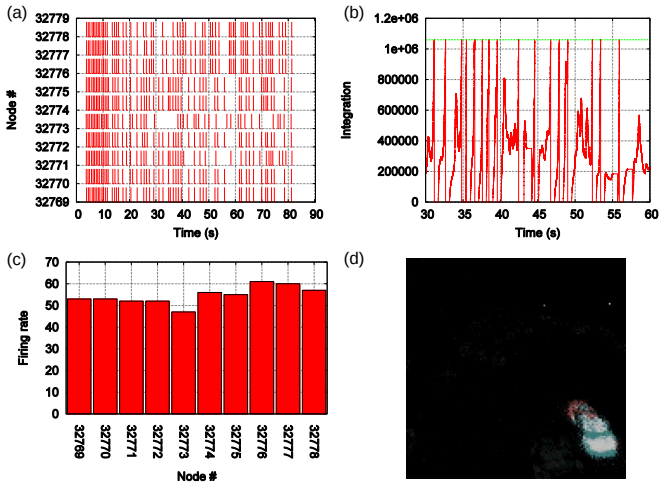

Fig. 5. Output reporting for a typical simulation. (a) Raster plot for 10 output nodes. (b) Internal integration state of a neuron. (c) Firing rate for 10 output nodes. (d) Weights reconstruction for an output neuron.

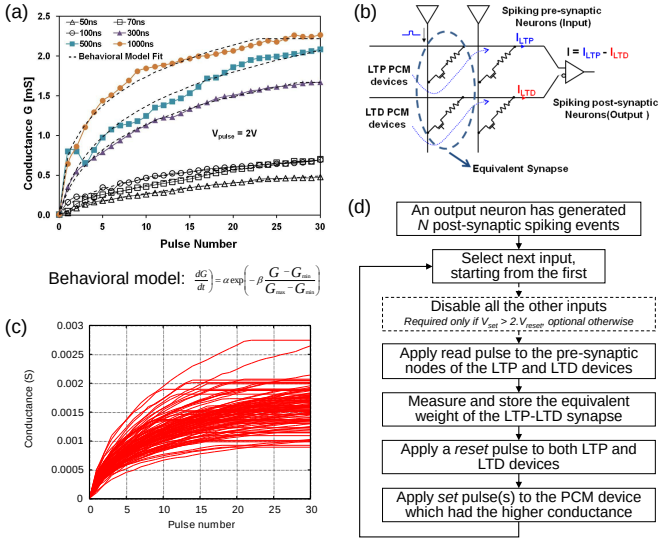

Fig. 6. Overview of a PCM-based synapse system modeling. (a) Experimental device data fitting (device's conductance increase vs. number of applied voltage pulses). (b) Two-PCM synapse modeling. (c) Simulated variability for the PCM devices over 100 potentiating cycles. (d) Reset operations diagram as implemented in Xnet.

Contrary to related work on PCM [28], the simulation proposed here avoid relying on an experimental STDP characterization, but rather uses directly the device behavior for simple potentiating pulses to implement STDP, which is done very easily thanks to the event-driven nature of Xnet. Experimentally measured device variability, modeled with normal or log-normal distribution, is therefore included at the device level and its impact on the STDP characteristic can be studied through Monte Carlo simulations (see next section).

This also offers the possibility to perform detailed synaptic learning analysis and estimate the total synaptic power consumption for a given technology, as shown in table I. The energy consumption is estimated by counting the total number of read, set and reset pulses during the learning. The energy of a pulse is estimated as $E \approx V \times I \times t$, where V, I and t are respectively the pulse voltage, current and duration. For the set and reset pulses, the current during phase change is essentially independent on the device conductance due to the threshold switching effect, and the average current measured experimentally is used. For the read pulses, the current is computed with Ohm's law according to the modeled device conductance.

TABLE I
LEARNING STATISTICS FOR THE PCM-BASED NEUROMORPHIC CIRCUIT, USED TO ESTIMATE THE SYNAPTIC POWER CONSUMPTION OF THE SYSTEM (FOR LANCE-TYPE GST-PCM TEST DEVICES, WITH A 100 NM THICK PHASE CHANGE LAYER AND 300 NM DIAMETER TUNGSTEN PLUG [26]).

Total Synapses: 1,966,680 [= 1st Layer (2×128×128×60) 2nd (60×10)]
Total PCM cells: 3,933,360 [= 2× Total Synapses (LTP and LTD)]
Total learning duration = 680 s

Quantity Values for	Number of pulses	Energy consumption
Total read pulses	4,975,830,080	0.12 pJ
Total set pulses	416,334,080	121 pJ
Total reset pulses	16,585,048	1552 pJ
Total synaptic power consumption		112 μW

D. Monte Carlo Simulations and Genetic Optimization of Parameters

Xnet includes facilities to perform distributed Monte Carlo simulations and genetic evolution on (synaptic and neural) network parameters, which are essential for variability robustness analysis and parameter space exploration. To this end, it implements a transparent, efficient and type-safe mechanism (derived from "Chameleon Objects" [29]) to declare any type of C++ variable as a model parameter that can be read and write into configuration files. Configuration files can be shared within a layer or distinct for each node of the network, allowing to load and save the entire state of the simulation. We use the standard Mersenne twister MT19937 pseudo-random number generator, which is optimized for Monte Carlo simulations [30].

In a first case study, Monte Carlo simulation is used to evaluate the robustness of the learning to variability on minimum (GOFF) and maximum (GON) synaptic device conductance. The results presented in figure 7 show the recognition rate for the MNIST handwritten digit database, in function of the relative standard deviation on GOFF and GON (for extrinsic, device to device variation and intrinsic, device variation during switching) [31]. In this experiment, each synapse is constituted of five binary resistive devices and the learning exploits the device switching stochasticity to implement STDP.

In a second case study, genetic evolution of neural parameter is used to evaluate the performance impact of system modifications and simplifications based on the setup presented in section III-B. The figure 8 illustrates how a new set of neural parameters can be found automatically through genetic evolution for a new system, with 4 bits digital synapses, linear neural leak instead of exponential and order-based STDP,

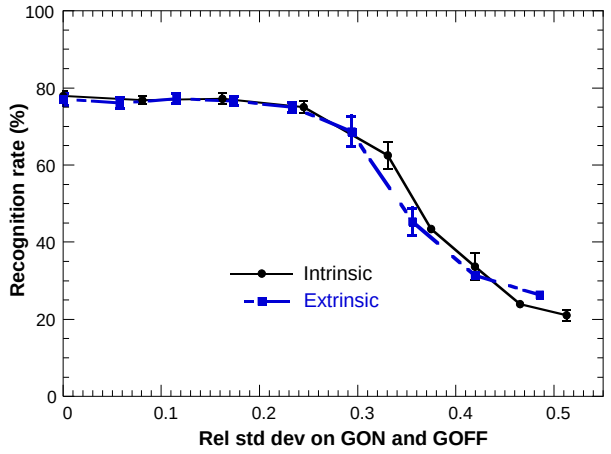

Fig. 7. Monte Carlo robustness analysis on digit recognition for intrinsic and extrinsic min. (GOFF) and max. (GON) synaptic conductance variability. Synapses are constituted of binary resistive devices and learning is stochastic [31].

where only the synapses activated by the latest 8192 events are potentiated upon activation of the output neuron [32].

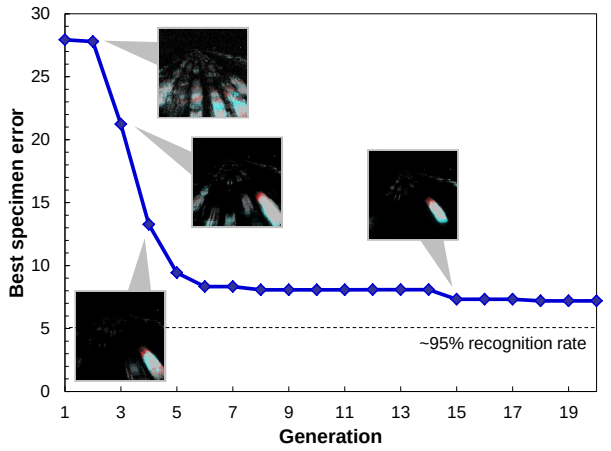

Fig. 8. Genetic evolution on neural parameters in Xnet to evaluate the performances of the simplified unsupervised features extraction system with 4 bits digital synapses, linear leak and order-based STDP [32]. The insets show the weights map of an output neuron after the learning of the car sequence, for four sets of parameter obtained at different point in the genetic evolution.

IV. Event-Based vs. Fixed Time-step Simulation

In the following, we compare the Xnet event-based simulation with the Brian fixed time-step one, in order to illustrate the main differences between the two types of simulation. We give some performance metrics, to help choosing the simulation type that is the most adapted depending on the parameters of the problem, regardless of the absolute figures obtained for this particular benchmark. Brian is used in its default configuration (according to the documentation, typical gains of 30% can be expected when activating compiled C code optimizations).

Xnet is not designed for biological systems modeling and simulation, contrary to neurosciences simulators, which are not designed to simulate neuromorphic hardware. The Brian simulator [14] however integrates more and more features towards this end. Using some Brian 1.4 experimental features, it is possible to emulate approximately the unsupervised features extraction presented earlier, from an AER recording. One limitation is that it does not allow to reset the membrane potential of the neurons through lateral inhibition directly, however this can be circumvented by disabling the pre-synaptic pulses integration during the inhibitory period, which allows the membrane potential to leak to zero.

The results in table II show that for 60 neurons ($\sim$2 millions synapses), Xnet is $\sim$400% faster than Brian with a default time step of 0.1 ms. In this simulation, each neuron is fully connected to the AER retina. The number of events is therefore proportional to the number of neurons, as each AER event is distributed to every neuron. With five times more neurons (300 neurons), the event-based simulation is as expected more than five times longer and uses five time more memory. As the number of events per time-step increases, clock-based simulation becomes more efficient and the Brian simulation is almost as fast as Xnet for 300 neurons. However, this can have side effects such as ineffective lateral inhibition, if multiple neurons fire in the same time-step, or poor precision during learning, if the time-step is several percents of the LTP or LTD windows. These issues are non-existent in event-based simulation, where the time precision is arbitrary (one femtosecond in Xnet). Parallelization of the event-driven priority queue is possible, although not implemented in Xnet, where code correctness and simplicity are preferred over simulation speed (which is not restrictive for our current needs). A one order of magnitude smaller time-step in Brian does not multiply the simulation time by ten, because synaptic updates are event-driven even in Brian and there is lateral inhibition in the simulated network.

The memory consumption is dominated by the synaptic parameters, which can be significant for semi-physical memristive device models. With a simple behavioral model, ten state variables are used for each synapse in Xnet, compared to only one state variable per connection in the equivalent Brian simulation, where synaptic variability is not modeled.

TABLE II
Xnet and Brian performances comparison. There are 1,966,080 synapses for N=60 neurons and 9,830,400 synapses for N=300 neurons. Xnet processes in average 2 million events/second.

Setup	Exec. time	Memory cons. (max)
Xnet (N=60)	131 s (×1.0)	344 MB (×1.0)
Brian (N=60, ΔT=0.1 ms)	524 s (×4.0)	399 MB (×1.2)
Brian (N=60, ΔT=0.01 ms)	1797 s (×5.2)	456 MB (×1.3)
Xnet (N=300)	817 s (×6.2)	1687 MB (×4.9)
Brian (N=300, ΔT=0.1 ms)	963 s (×7.4)	1134 MB (×3.3)
Brian (N=300, ΔT=0.01 ms)	2157 s (×17)	1192 MB (×3.5)
Xnet (N=1500)	5693 s (×43)	8224 MB (×24)
Brian (N=1500, ΔT=0.1 ms)	2826 s (×22)	4823 MB (×14)
Brian (N=1500, ΔT=0.01 ms)	5168 s (×39)	4881 MB (×14)

V. Conclusion

Xnet was developed to allow fast and efficient design exploration of spiking neuromorphic architectures by providing

a framework that can mix high-level behavioral modeling with hardware constraints integration. More specifically, it was designed with spiking retina and memristive nano-devices based architectures in mind. The Xnet source code is currently not licensed, but is available through partnership with CEA LIST. Additional information on implementation details is available upon request.

This simulator, along with its accompanying methodology was used for the simulations in several papers published by our group [8], [25]–[27] and is now mature and useful to be introduced to the community, in addition to the necessity to provide means of reproducing and validating scientific results.

Until now, the neuromorphic hardware community lacked a generic and efficient framework for high-level memristor-based spiking neuromorphic architectures exploration. We hope that Xnet will be a first step towards such a framework, as it proved to be extremely valuable for rapid evaluation of new nanodevices and for a better understanding of STDP-like learning rules. Xnet is always under active development, with current integration of magnetic tunnel junction synaptic device model and audio processing and learning simulations.

REFERENCES

[1] S. B. Furber, S. Temple, and A. D. Brown, "High-performance computing for systems of spiking neurons," *University of Pennsylvania Law Review*, vol. 154, no. 3, pp. 477+, 2006.

[2] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 2008, pp. 431–438.

[3] Choudhary, S. Sloan, S. Fok, A. Neckar, E. Trautmann, P. Gao, T. Stewart, C. Eliasmith, and K. Boahen, "Silicon neurons that compute," in *International Conference on Artificial Neural Networks*, vol. 7552, 2012, pp. 121–28.

[4] J. Arthur, P. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. Esser, N. Imam, W. Risk, D. Rubin, R. Manohar, and D. Modha, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, pp. 1–8.

[5] G. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *Nanoscale Architectures, 2008. NANOARCH 2008. IEEE International Symposium on*, 2008, pp. 85–92.

[6] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[7] S. Yu and H.-S. Wong, "Modeling the switching dynamics of programmable-metallization-cell (PMC) memory and its application as synapse device for a neuromorphic computation system," in *Electron Devices Meeting (IEDM), 2010 IEEE International*, 2010, pp. 22.1.1–22.1.4.

[8] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011, pp. 1775–1781.

[9] C. Zamarreño-Ramos, L. A. Camuñas-Mesa, J. A. Perez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," *Frontiers in Neuroscience*, vol. 5, no. 26, 2011.

[10] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, no. 73, 2011.

[11] *VHDL Language Reference Manual*, IEEE Std. 1076-2008, 2009.

[12] *SystemC Language Reference Manual*, IEEE Std. 1666-2005, 2006.

[13] N. Carnevale and M. Hines, *The NEURON Book*. New York, NY, USA: Cambridge University Press, 2006.

[14] D. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in python," *Frontiers in Neuroinformatics*, vol. 2, no. 5, 2008.

[15] M.-O. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[16] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.

[17] M. Suri, O. Bichler, D. Querlioz, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo, "CBRAM devices as binary synapses for low-power stochastic neuromorphic systems: Auditory (cochlea) and visual (retina) cognitive processing applications," in *Electron Devices Meeting (IEDM), 2012 IEEE International*, 2012, in press.

[18] *C++11*, ISO/IEC Std. 14 882:2011, 2011.

[19] A. Delorme, J. Gautrais, R. van Rullen, and S. Thorpe, "SpikeNET: A simulator for modeling large networks of integrate and fire neurons," *Neurocomputing*, vol. 26, pp. 989–996, 1999.

[20] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 2, pp. 566–576, 2008.

[21] S.-C. Liu, A. van Schaik, B. Minch, and T. Delbruck, "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010, pp. 2027–2030.

[22] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *Nanotechnology, IEEE Transactions on*, 2013, in press.

[23] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010, pp. 2426–2429.

[24] D. Querlioz, P. Dollfus, O. Bichler, and C. Gamrat, "Learning with memristive devices: How should we model their behavior?" in *Nanoscale Architectures (NANOARCH), 2011 IEEE/ACM International Symposium on*, 2011, pp. 150 –156.

[25] O. Bichler, D. Querlioz, S. Thorpe, J. Bourgoin, and C. Gamrat, "Unsupervised features extraction from asynchronous silicon retina through spike-timing-dependent plasticity," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011, pp. 859–866.

[26] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *Electron Devices Meeting (IEDM), 2011 IEEE International*, 2011.

[27] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, and C. Gamrat, "Visual pattern extraction using energy-efficient '2-PCM synapse' neuromorphic architecture," *Electron Devices, IEEE Transactions on*, vol. 59, no. 8, pp. 2206–2214, 2012.

[28] D. Kuzum, R. Jeyasingh, S. Yu, and H.-S. Wong, "Low-energy robust neuromorphic computation using synaptic devices," *Electron Devices, IEEE Transactions on*, vol. 59, no. 12, pp. 3489–3494, 2012.

[29] V. Simonis and R. Weiss, "Heterogeneous, nested STL containers in C++," in *Perspectives of System Informatics*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, vol. 1755, pp. 383–388.

[30] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998.

[31] D. Querlioz, O. Bichler, M. Suri, J. Larroque, Z. W.S., J.-O. Klein, S. Retailleau, P. Dollfus, B. De Salvo, and C. Gamrat, "Simulation of a bio-inspired system capable of learning thanks to stochastic nanodevices," *Applied Physics Letters*, 2013, (submitted).

[32] D. Roclin, O. Bichler, C. Gamrat, S. J. Thorpe, and J.-O. Klein, "Design study of an efficient digital order-based STDP neuron implementation for temporal features extraction," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, 2013, (accepted).