Mapping arbitrary mathematical functions and dynamical systems to neuromorphic VLSI circuits for spike-based neural computation

Federico Corradi*, Chris Eliasmith†, and Giacomo Indiveri*

*Institute of Neuroinformatics, University of Zürich and ETH Zürich, Switzerland
†Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON, Canada

Email: {federico,giacomo}@ini.phys.ethz.ch

***Abstract*—Brain-inspired, spike-based computation in electronic systems is being investigated for developing alternative, non-conventional computing technologies. The Neural Engineering Framework provides a method for programming these devices to implement computation. In this paper we apply this approach to perform arbitrary mathematical computation using a mixed signal analog/digital neuromorphic multi-neuron VLSI chip. This is achieved by means of a network of spiking neurons with multiple weighted connections. The synaptic weights are stored in a 4-bit on-chip programmable SRAM block. We propose a parallel event-based method for calibrating appropriately the synaptic weights and demonstrate the method by encoding and decoding arbitrary mathematical functions, and by implementing dynamical systems via recurrent connections.**

I. Introduction

Threatened by the approaching limits of Moore's law [1], semiconductor industries and research labs started investigating alternative signal processing and computational approaches for developing new generations of computing technologies that can go beyond standard Complementary Metal–Oxide–Semiconductor (CMOS) solutions [2]. One promising approach is that of implementing brain-inspired models of neural computation, based on massively parallel networks of low-power silicon neuron circuits [3]. Within this context, several promising devices have recently proposed, using both digital and analog design techniques [4]–[9]. However, in order to implement full-fledged computing systems, starting from these types of devices, it is necessary to adopt a formalism that can best exploit the properties of such computing elements. The Neural Engineering Framework (NEF) [10] represents a synthesis of multiple approaches in computational neuroscience, computer science, communications and control theory, that can provide such formalism. This computational framework has been previously introduced in [10] and it has been used to simulate in Software (SW) large spiking neural networks capable of reproducing many aspects of neural systems ranging from the neurophysiological level all the way up to the behavioural one [11].

In this paper we validate the NEF by applying its principles to a population of compact low-power silicon neurons, designed using neuromorphic analog circuits and fabricated using a standard $0.18\,\mu$m CMOS process. We provide experimental results showing the outcome of the calibration procedures required to implement NEF, and of a successful real-time computation of mathematical functions. In addition, using this framework, we construct a dynamical system with two distributed memory states that represent the neural correlate of working memory, and demonstrate its correct real-time performance in Hardware (HW).

II. Materials and Methods

The Very Large Scale Integration (VLSI) device used in this work is a prototype chip that comprises a network of 58 adaptive exponential Integrate-and-Fire (I&F) neurons, implemented using analog subthreshold circuits. Each neuron has 32 programmable synaptic inputs, with synapse circuits that express biologically plausible neural dynamics. In addition, each neuron has 8 bi-stable synapses, with on-chip plasticity mechanisms. The VLSI chip can receive and transmit pulses representing spikes via asynchronous digital circuits, and following an Address Event Representation (AER) protocol. The chip is connected to a workstation via a USB interface; signals transmitted to the USB bus from the chip encode the address of the source neuron, while signals received by the chip encode the address of the destination synapse. Once off-chip, the spikes produced by the silicon neurons are routed by a "mapper" board, built using a commercial FPGA (Xilinx Spartan-6). The mapper is hosted in a standard workstation and uses the workstation memory to implement a programmable connectivity look-up table with source-destination entries. This setup allows us to construct arbitrary network topologies of spiking neural networks.

A. Principles of the framework

The neural engineering framework (NEF) [10] is based on control theory and integrates three basic principles:

1) Representation: a stimulus $x(t)$ is encoded as spiking activity $a_i(x(t))$ by a pool of neurons with different transfer characteristics: $a_i(x(t)) = G_i(\alpha_i e_i x(t) + J_i^{bias})$ where G_i is a

spiking neural nonlinearity, α is a gain, e is an encoder, and J_i^{bias} is a background current. This activity $a_i(x(t))$ is linearly decoded to retrieve the stimulus $x(t)$, $\hat{x}(t) = \sum_i a_i(x(t))d_i^x$ in which d_i^x represents the decoding weights, and $a_i(t) = \sum_n h_{PSC}(t) * \delta(t - t_{in})$ is the linearly filtered spiking activity. The $h_{PSC}(t)$ is a filter capturing the effects of postsynaptic currents.

2) Transformation: a stimulus $x(t)$ is transformed into $y(t)$ by a mapping of $a_i(x(t))$ into $b_j(y(t))$. The transformation is a weighted connection between neural pools of neurons that compute a function on the represented value. For example, in the linear case $y(t) = Ax(t)$ is represented by the activity $b_j(A\hat{x}(t))$. In this representation neuron i feeds its output to the input of neuron j by using a weight matrix $\omega_{ij} = \alpha_j e_j A d_i$.

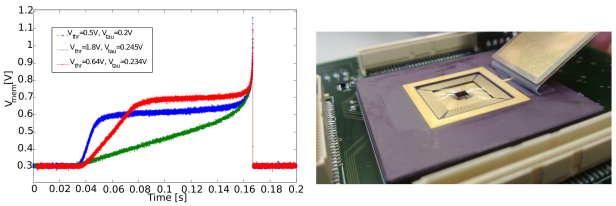
3) Dynamics: recurrent connections can be computed using the same approach to implement nonlinear and linear dynamical models including attractor networks, Kalman filters, controllable harmonic oscillators, etc. In such a mapping, the standard dynamics matrix A becomes $A' = \tau_{PSC}A + I$, and the input matrix B becomes $B' = \tau_{PSC}B$ where τ_{PSC} is the time constant of the postsynaptic current filter.

B. The VLSI realization

We applied the NEF to the multi-neuron chip, exploiting its analog silicon neuron properties, and its programmable synaptic weight features.

1) The silicon neuron: The silicon neurons on this chip implement models of adaptive exponential I&F neuron [3]. Figure 1 shows the membrane potential of one of these neurons, excited by constant current in the order of few pA. The three traces in the plot represent the membrane potential for three different bias threshold voltages V_{thr} and integration time constants V_{tau}. The neuron circuit is shown in Fig. 2. The synaptic input current I_{syn} is low-pass filtered by a Differential Pair Integrator (DPI) filter [12] (see Fig.2A), which includes an adjustable threshold voltage transistor V_{thr} and a leak conductance bias V_{tau}. The membrane capacitance integrates the input current and generates the membrane potential V_{mem}. A positive-feedback inverting amplifier is used to generate spike events with very low-power consumption [3] (see Fig. 2C). Spike frequency adaptation is implemented by an additional DPI circuit in negative feedback configuration (Fig. 2B). The magnitude of the adaptation current can be controlled via the bias voltage V_{adw}, while the time constant is controlled by the voltage bias V_{adtau}. Fig. 1D contains the spike communication inverter (M_{D4}, M_{D3}), the refractory period bias V_{refr} transistor and the bias that regulates the reset voltage V_{reset} of the neuron after a spike event.

2) The asynchronous programmable Static Random Access Memory (SRAM) synapses: Synaptic weights are stored with 4-bit resolution in a standard 10T SRAM block [13] integrated on the same chip with an asynchronous interface as described in [4]. The digital weight values are converted into an analog current by a Digital to Analog Converter (DAC) circuit integrated in the DPI synapse circuits (see Fig. 3). The bias voltages W_0, W_1, W_2, and W_3 are used to weight the



(a) (b)

Fig. 1: (a). Membrate potential for different bias voltage parameters. (b). Photo of the neuromorphic chip.

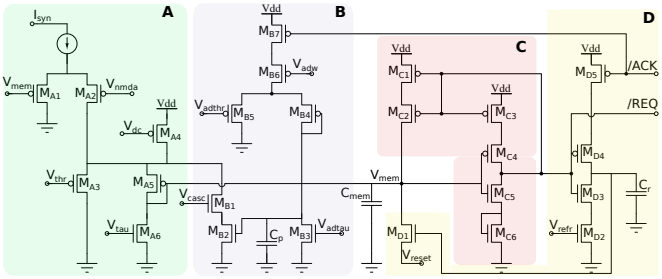


Fig. 2: Adaptive exponential I&F neuron circuit schematic. (a) Current input block. (b) Adaptation circuit. (c) Positive-feedback inverting amplifier. (d) Digital communication and reset block.

single bit values. The *PU*-biased p-Field Effect Transistor (FET) is necessary to eliminate charge-pump effects. The DAC output current $I_{synSRAM}$ determines the gain of the excitatory or inhibitory DPI synapse. The excitatory (inhibitory) synapse produces an exponential Excitatory Post Synaptic Current (EPSC) (Inhibitory Post-Synaptic Current (IPSC)) which emulates the exponential ligand-gated postsynaptic current generation mechanism [12].

III. Results

A. Weight calibration using SRAM synapses

The asynchronous SRAM synapses have been used to compensate for device mismatch caused by the process variations.

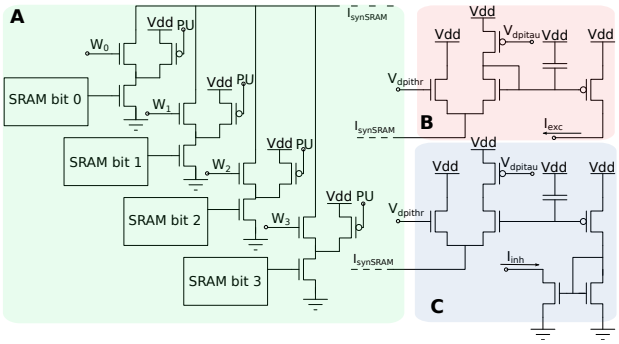
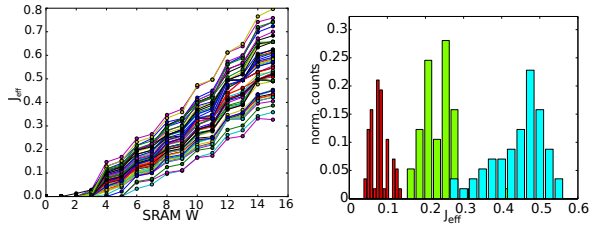
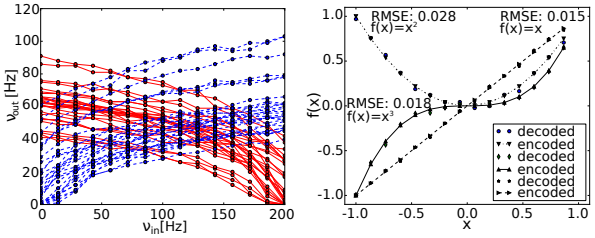


Fig. 3: Programmable synapse circuit. (a) SRAM digital to analog converter block. (b) Excitatory DPI synapse. (c) Inhibitory DPI synapse.



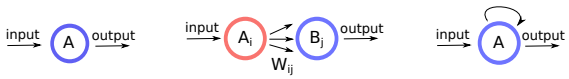
(a) (b)

Fig. 4: SRAM calibration. (a) Synaptic efficacy (J_{eff}) for different sram weight values. Every line represents a neuron in the array. The synaptic efficacy J is expressed in terms of number of excitation spikes over spike emitted by the neuron under exam. (b) Calibrated SRAM weight to match three different synaptic efficacies.



(a) (b)

Fig. 5: (a) Tuning functions for all neurons. (b) Encoding and decoding functions. $f(x) = x$ $RMSE = 0.0155$, $f(x) = x^2$ $RMSE = 0.0287$, $f(x) = x^3$ $RMSE = 0.0187$.



(a) (b) (c)

Fig. 6: Schematic diagrams. (a) Encoding with neurons. (b) Transformations. (c) Integrator.

One of the main problems, when using analog neuromorphic systems, is the inability to obtain precise synaptic weights. We overcome this problem by applying a fast calibration procedure that considers the response function of single neuron by adjusting the digital weight stored in SRAM cells to match for a chosen synaptic efficacy value. The calibration procedure consists of measuring effective synaptic efficacies for all neurons in the array and for all possible digital weights. This procedure can be executed in parallel for all neurons in the chip and it only requires to send (receive) spikes to (from) the chip using the Address-Event Representation. Fig. 4 shows the result of this measure when all neurons are stimulated at $100Hz$ for $200ms$. The stimulation is done via synthetic regular spike trains produced by the computer. The complete calibration only takes $200ms * 15(bits) = 3s$. Once we measured all calibration curves, we use the least squares estimates method to estimate a digital value for a given synaptic efficacy. With this method, we are capable of obtaining synaptic efficacies as shown in Fig. 7.

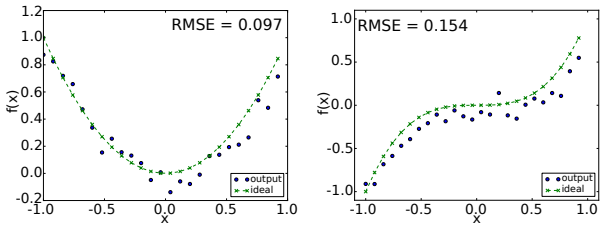
B. Representations of functions with populations of neurons

In general, encoding is obtained by the spiking activity $\delta(t - t_{in})$ of a single neuron i via the nonlinear neuron response function G_i. In practice, encoding thus exploits different tuning curves for neurons that project the stimulus $x(t)$ to a specific neuron space. In fact, tuning curves relates the spiking response of a neuron to a particular stimulus. We show in Fig. 5a tuning curves for all neurons in the neuromorphic chip. We implemented different bias values J_i^{bias} by stimulating every neuron i with a fixed Gaussian spike train. The mean of the Gaussian spike train is picked from a flat random distribution between $10Hz$ and $80Hz$. Encoders values are randomly picked between two alternatives $(+1,-1)$, this gives the different directions of the tuning curves in Fig. 5a. Example of encoded and decoded mathematical functions are shown in Fig. 5b. The protocol of this experiment is the stimulation of all neurons with a swept Poisson spike train from $1Hz$, to $200Hz$ in 25 steps of $200ms$. The optimal linear decoders, d_i, used in Fig. 5b, are estimated by minimizing the expression $< (x - \hat{x})^2 >_x$ with the least square method under some expected mean zero, independent Gaussian noise.

C. Real-time computations of mathematical functions

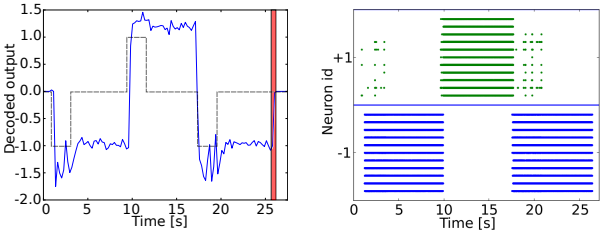
Mathematical computation can be performed by pools of neurons in which the encoded representations are defined by $a_i = G_i[\alpha_i \langle x \cdot e_i \rangle + J_i^{bias}], b_j = G_j[\alpha_j \langle y \cdot e_j \rangle + J_j^{bias}]$, and the respective representational decodings are: $\hat{x} = \sum_i a_i d_i, \hat{y} = \sum_i b_j d_j$. Note that it is possible to find optimal decoders for arbitrary nonlinear functions of x using this same technique. We denote these as $d_i^{f(x)}$. It is thus possible to compute the desired mathematical computation, by substituting estimates of the desired function into b such that $y = f(x) \approx \hat{f}(x)$ we obtain $b_j = G_i \left[\sum_i \omega_{ji} a_i + J_i^{bias}\right]$, in which the weights matrix is $\omega_{ji} = a_j e_j d_i^{f(x)}$. The schematic representations in terms of pool of neurons is shown in Fig. 6b. We used 28 neurons for each population (A, B). All to all connections are realized using the mapper board. We achieved computation as shown in Fig. 7. Neurons in population A are excited with a ramping Poisson spike train from $1Hz$ to $200Hz$ in step of $200ms$. This is equivalent to the input range $[-1,1]$. Population A spiking activity is in real-time directed to population B whose output is the desired computed mathematical function.

1) Working memory as a dynamical system: We implemented a stable dynamical system by introducing recurrent connections in the network. We realized the neural correlates of working memory, this means that the network dynamics is capable of storing input values through self-sustained activity. To achieve memory states, we implemented the dynamics of an integrator, Fig.6c. The third principle of NEF describes the relationship between standard control theory and neural dynamics, in this mapping an integrator is described by the transformation matrices in which $A' = 1$ and $B' = \tau_{PSC}$. Moreover, if we assume an exponential postsynaptic current



(a) (b)

Fig. 7: Computing mathematical functions with neurons. (a) computed function $f(x) = x^2$, $RMSE = 0.097$. (b) computed function $f(x) = x^3$, $RMSE = 0.154$.



(a) (b)

Fig. 8: Dynamical system: integrator. (a) Decoded activity from the pool of neurons recurrently connected. The dotted line represents input to the network, note that when input is removed the network stores its integrated value in an attractor state. The red stripe, $t = 26s$, defines an inhibitory input stimulus sent via the computer. (b) Raster plot of network activity, the two colors (blue, green) represent neurons with different encoders (+1 or -1).

($h_{PSC}(t) = \frac{1}{\tau} e^{\frac{-t}{\tau}}$) and a linear time-invariant system, recurrent weights can be computed from the dynamical matrices of the system. We used a pool of 22 neurons recurrently connected. Encoder values where randomly picked between two alternatives (+1,−1). In Fig.8 we show the activity over time of the integrator. Fig.8a shows in dotted line the input generated by the computer and in continuous line the decoded output value. At first, we excite the neural pool with an input directed to the neuron encoding for -1, and at the removal of the input, the network successfully store -1 value in a reverberant state of activity. At $t = 10s$, we excite the the network with a $+1$ stimulus and the network correctly stores it. After ad additional storage of -1 ($t = 16s$), we kill the activity of the network at $t = 26s$ with an inhibitory input stimulus. Fig.8b shows the raster plot of the run.

IV. Conclusions

This work addresses the challenge of obtaining distributed and programmable computation with noisy and heterogeneous analog circuits in a network of spiking neurons. We demonstrated that arbitrary computation in a neuromorphic multi-neuron VLSI chip can be achieved using the NEF's principles.

The NEF framework represents a robust method for computing connection weights that takes into account neurons with a wide range of different transfer characteristic. This requirement makes the framework appealing to analog computation, as the effect of device mismatch is compatible with the diversity requirements, and allows for very compact neuron designs. On the other hand, the method requires precise synaptic weights. To compensate for the negative effects of mismatch in the neuron and synapse circuits, we exploited the availability of programmable SRAM cells, while keeping the synapse and neuron circuits compact. As demonstration, we showed reliable computation of functions across different pools of neurons. Additionally, we constructed robust dynamic attractor states by introducing recurrent connections in the network. Such stable dynamical systems are fundamental for building complex neural systems, and developing brain inspired spike-based computing systems.

Acknowledgment

This work has been supported by the European Commission with the "neuroP" Project, ERC-2010-StG-257219-neuroP.

References

[1] B. Hoefflinger, "Itrs: The international technology roadmap for semiconductors," in *Chips 2020*, ser. The Frontiers Collection, B. Hoefflinger, Ed. Springer Berlin Heidelberg, 2012, pp. 161–174. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23096-7_7 1

[2] R. Cavin, P. Lugli, and V. Zhirnov, "Science and engineering beyond moore's law," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1720–1749, 2012. 1

[3] G. Indiveri, *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, pp. 1–23, 2011. [Online]. Available: http://www.frontiersin.org/Neuromorphic_Engineering/10.3389/fnins.2011.00073/abstract 1, 2

[4] S. Moradi and G. Indiveri, "An event-based neural network architecture with an asynchronous programmable synaptic memory," *IEEE Transactions on Biomedical Circuits and Systems*, March 2013. 1, 2

[5] A. Cassidy, J. Georgiou, and A. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Networks*, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608013001597 1

[6] J. Arthur, *et al.*, "Building block of a programmable neuromorphic substrate: A digital neurosynaptic core," in *International Joint Conference on Neural Networks, IJCNN 2012*. IEEE, Jun 2012, pp. 1946–1953. 1

[7] S. Choudhary, S. Sloan, S. Fok, A. Neckar, E. Trautmann, P. Gao, T. Stewart, C. Eliasmith, and K. Boahen, "Silicon neurons that compute," in *Artificial Neural Networks and Machine Learning – ICANN 2012*, ser. Lecture Notes in Computer Science, A. Villa, W. Duch, P. Érdi, F. Masulli, and G. Palm, Eds. Springer Berlin / Heidelberg, 2012, vol. 7552, pp. 121–128. 1

[8] S. Brink, S. Nease, and P. Hasler, "Computing with networks of spiking neurons on a biophysically motivated floating-gate based neuromorphic integrated circuit," *Neural Networks*, 2013. 1

[9] T. Yu, J. Park, S. Joshi, C. Maier, and G. Cauwenberghs, "65k-neuron integrate-and-fire array transceiver with address-event reconfigurable synaptic routing," in *Biomedical Circuits and Systems Conference, (BioCAS), 2012*. IEEE, Nov. 2012, pp. 21–24. 1

[10] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation and dynamics in neurobiological systems*. Cambridge, MA: MIT Press, 2003. 1

[11] C. Eliasmith, T. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012. [Online]. Available: http://www.sciencemag.org/content/338/6111/1202.abstract 1

[12] C. Bartolozzi and G. Indiveri, "Synaptic dynamics in analog VLSI," *Neural Computation*, vol. 19, no. 10, pp. 2581–2603, Oct 2007. [Online]. Available: http://ncs.ethz.ch/pubs/pdf/Bartolozzi_Indiveri07.pdf 2

[13] C.-H. Lo and S.-Y. Huang, "Ppn based 10t sram cell for low-leakage and resilient subthreshold operation," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 3, pp. 695–704, 2011. 2