Silicon Neurons That Compute

Swadesh Choudhary, Steven Sloan, Sam Fok, Alexander Neckar,
Eric Trautmann, Peiran Gao, Terry Stewart,
Chris Eliasmith, and Kwabena Boahen

Stanford University,
Stanford, CA 94305, U.S.A.
{swadesh,ssloan1,samfok,aneckar,etraut,prgao,boahen}@stanford.edu,
{tcstewart,celiasmith}@uwaterloo.ca

Abstract. We use neuromorphic chips to perform arbitrary mathematical computations for the first time. Static and dynamic computations are realized with heterogeneous spiking silicon neurons by programming their weighted connections. Using 4K neurons with 16M feed-forward or recurrent synaptic connections, formed by 256K local arbors, we communicate a scalar stimulus, quadratically transform its value, and compute its time integral. Our approach provides a promising alternative for extremely power-constrained embedded controllers, such as fully implantable neuroprosthetic decoders.

Keywords: Neuromorphic chips, Silicon neurons, Probabilisitic synapses.

1 Brain-Inspired Analog–Digital Systems

Analog computation promises extreme energy-efficiency by operating close to the shot-noise limit [1]. By exploiting physical laws (e.g., conservation of charge for summation), a handful of analog devices is sufficient to perform computation. In contrast, digital computation relies on abstractions that require many more devices to achieve the same function (e.g., hundreds of transistors to add two 8-bit numbers). Furthermore, these abstractions break when noise exceeds a critical level, requiring enormous noise margins to avoid catastrophic failures. In contrast, analog degrades gracefully, allowing for operation at low noise margins, thereby saving power.

However, robust and programmable computation using noisy analog circuits is challenging. Robust computation requires a distributed approach, but this is difficult because analog communication is susceptible to heterogeneity and noise. Programmable computation requires flexibility, but this is also difficult because analog computation exploits the underlying devices' fixed physical properties.

In this paper, we realize robust and programmable mathematical computations with noisy and heterogeneous components using a framework inspired by the brain [2]. The brain uses graded dendritic potentials (cf. analog computation), all-or-none axonal spikes (cf. digital communication) and probabilistic synapses (cf. weighted connections). Our analog computational units are spiking silicon neurons [5]; our digital communication fabric is a packet-switched

A.E.P. Villa et al. (Eds.): ICANN 2012, Part I, LNCS 7552, pp. 121–128, 2012.

network [3]; and our weighted connections are packet-delivery probabilities [8]. While neuromorphic systems that combine some or all of these features of the brain have been built previously, they only performed specific computations. In contrast, we realize any desired mathematical computation by programming the connections' weights to exploit the silicon neurons' heterogeneity.

Section 2 reviews a theoretical framework for mapping computations onto heterogeneous populations of spiking neurons. Section 3 presents hardware elements that support this framework. Section 4 describes *Neurogrid*, a sixteen-chip, million-neuron neuromorphic system used as a testbed (proposed in [4]). Section 5 presents implementations of static and dynamic computations. Section 6 discusses possible applications.

2 The Neural Engineering Framework

To program the connection weights among heterogeneous spiking neurons (indexed by i), NEF follows three principles [2] (Figure 1):

Representation: A multi-dimensional stimulus $\mathbf{x}(t)$ is nonlinearly encoded as a spike rate $a_i(\mathbf{x}(t))$—represented by the *neuron tuning curve*—that is linearly decoded to recover an estimate of $\mathbf{x}(t)$, $\hat{\mathbf{x}}(\mathbf{t}) = \sum_i a_i(\mathbf{x}(t))\phi_i^{\mathbf{x}}$, where $\phi_i^{\mathbf{x}}$ are the decoding weights.

Transformation: Transformations of $\mathbf{x}(t)$ into $\mathbf{y}(t)$ are mapped directly to transformations of $a_i(\mathbf{x}(t))$ into $b_j(\mathbf{y}(t))$ using alternate decoding weights. For example, $\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t)$ is represented by the spike rates $b_j(\mathbf{A}\hat{\mathbf{x}}(t))$, where neuron j's input is computed directly from neuron i's output using $\mathbf{A}\hat{\mathbf{x}}(t) = \sum_i a_i(\mathbf{x}(t))\mathbf{A}\phi_i^{\mathbf{x}}$, an alternative linear weighting.

Dynamics: Dynamics are realized using the synapses' spike response $h(t)$. This principle brings together the first two principles and adds the time dimension. For example, for $h(t) = \tau^{-1}e^{-t/\tau}$, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{y}(t)$ is realized as the equivalent, *neurally plausible* dynamical system: $\mathbf{x}(t) = h(t) * (\mathbf{A}'\mathbf{x}(t) + \mathbf{B}'\mathbf{y}(t))$, where convolution replaces integration, $\mathbf{A}' = \tau\mathbf{A} + \mathbf{I}$, and $\mathbf{B}' = \tau\mathbf{B}$.

The first principle is realized by assigning neuron i a randomly chosen preferred direction in the stimulus space, $\tilde{\phi}_i$:

$$a_i(\mathbf{x}(t)) = G(J_i(\mathbf{x}(t))) \text{ where } J_i(\mathbf{x}(t)) = \alpha_i\langle\tilde{\phi}_i \cdot \mathbf{x}(t)\rangle + J_i^{\text{bias}} \tag{1}$$

Here G is the neurons' nonlinear current-to-spike-rate function. The dot product converts the multi-dimensional stimulus, $\mathbf{x}(t)$, to a one-dimensional soma current, J_i. α_i is a gain or conversion factor and J_i^{bias} is a bias current. These two parameters are chosen to uniformly distribute firing thresholds and maximum firing rates within specified ranges (Figure 1, *left*). For a one-dimensional (1D) stimulus space, $\tilde{\phi}_i = \pm 1$. In contrast, the linear decoding weights, $\phi_i^{\mathbf{x}}$, are obtained by minimizing the mean square error. This error may be computed relative to the original stimulus $\mathbf{x}(t)$ or some nonlinear function thereof, $\mathbf{f}(\mathbf{x}(t))$, yielding $\hat{\mathbf{f}}(\mathbf{x}(t)) = \sum_i a_i(\mathbf{x}(t))\phi_i^{\mathbf{f}(\mathbf{x})}$.