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.

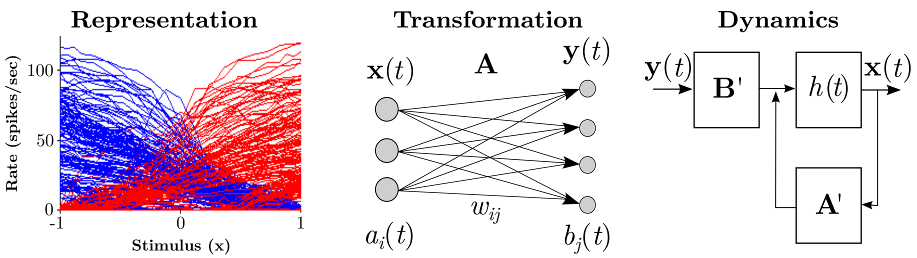
Transformation: Transformations of $\mathbf{x}(t)$ into $\mathbf{y}(t)$ are mapped directly to transformations of $a_i(\mathbf{x}(t))$ into $b_j(\mathbf{y}(t))$ using alternate decoding weights. For example, $\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t)$ is represented by the spike rates $b_j(\mathbf{A}\hat{\mathbf{x}}(t))$, 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 $\hat{\mathbf{f}}(\mathbf{x}(t)) = \sum_i a_i(\mathbf{x}(t))\phi_i^{\mathbf{f}(\mathbf{x})}$.



$$\mathbf{x} \rightarrow a_i(\mathbf{x}) \text{ and } \hat{\mathbf{x}} = \sum_i a_i(\mathbf{x})\phi_i^{\mathbf{x}} \qquad w_{ij} = \langle \tilde{\phi}_j \cdot \mathbf{A}\phi_i^{\mathbf{f(x)}} \rangle \qquad \mathbf{x} = h * (\mathbf{A}'\mathbf{x} + \mathbf{B}'\mathbf{y})$$

$$a_i(\mathbf{x}) = G(\alpha_i \langle \tilde{\phi}_i \cdot \mathbf{x} \rangle + J_i^{\text{bias}}) \quad b_j = G(\alpha_j \textstyle\sum_i w_{ij} a_i + J_j^{\text{bias}}) \quad \mathbf{A}' = \tau\mathbf{A} + \mathbf{I},\, \mathbf{B}' = \tau\mathbf{B}$$

Fig. 1. NEF's three principles. **Representation:** Tuning curves map stimuli ($\mathbf{x}(t)$) to spike rates ($a_i(\mathbf{x}(t))$). **Transformation:** Populations $a_i(t)$ and $b_j(t)$ connected by weights w_{ij} transform $\mathbf{x}(t)$'s representation into $\mathbf{y}(t)$'s. **Dynamics:** Synapses' spike response, $h(t)$, implement dynamics using neurally plausible matrices $\mathbf{A}'$ and $\mathbf{B}'$.

The second and third principles are realized by using the matrix $\mathbf{A}$ that describes $\mathbf{x}(t)$'s transformation or dynamics to specify the weight w_{ij} connecting neuron i to neuron j [2]:

$$w_{ij} = \langle \tilde{\phi}_j \cdot \mathbf{A}\phi_i^{\mathbf{f(x)}} \rangle \Rightarrow J_j = \alpha_j \sum_i w_{ij} a_i(\mathbf{x}(t)) + J_j^{\text{bias}} = \langle \tilde{\phi}_j \cdot \mathbf{A}\hat{\mathbf{f}}(\mathbf{x}(t)) \rangle + J_j^{\text{bias}} \quad (2)$$

where $\phi_i^{\mathbf{f(x)}}$ is neuron i's decoding vector and $\tilde{\phi}_j$ is neuron j's encoding vector (Figure 1, *middle*, *right*). Neuron j will fire as if it received an input of $\mathbf{y}(t) = \mathbf{A}\hat{\mathbf{f}}(\mathbf{x}(t))$; hence decoding its firing rate will yield the desired result. This recipe enables a computation specified in a low-dimensional space ($\mathbf{A}$) to be solved in a high-dimensional space (w_{ij}) using lower precision elements.

3 Hardware Elements

To support NEF in hardware, we need spiking silicon neurons, exponentially decaying synapses and programmable interconnection weights. Neurogrid implements these elements as follows.

Silicon neurons are implemented with quadratic integrate-and-fire dynamics:

$$\tau_{\text{m}}\dot{v} = -v + v^2/2 + g_{\text{syn}}(e_{\text{rev}} - v) \quad (3)$$

where τ_{m} is the membrane time-constant, v is the membrane potential (normalized by the threshold voltage), g_{syn} is the total synaptic conductance (normalized by the leak conductance), and e_{rev} is the reversal potential (also normalized by the threshold voltage); v is reset to 0 when it exceeds 10. Integration yields the inter-spike intervals and inversion yields the conductance-to-spike-rate function:

$$G(g_{\text{syn}}) = \left(\tau_{\text{m}} \frac{\pi + 2\,\text{arccot}(\sqrt{2e_{\text{rev}}\, g_{\text{syn}}/(1+g_{\text{syn}})^2 - 1})}{(1+g_{\text{syn}})\sqrt{2e_{\text{rev}}\, g_{\text{syn}}/(1+g_{\text{syn}})^2 - 1}} + t_{\text{ref}} \right)^{-1} \quad (4)$$

where t_{ref} is the refractory period [6]. This nonlinear function is appropriate as NEF does not favor any particular neuron model.

Silicon synapses are implemented by low-pass filtering unit-amplitude digital pulses, $p_{\text{rise}}(t)$, of width t_{rise} [7]:

$$\tau_{\text{syn}}\,\dot{g}_{\text{syn}} = -g_{\text{syn}} + g_{\text{sat}}\min(\textstyle\sum_i p_{\text{rise}}(t - t_i), 1) \tag{5}$$

where τ_{syn} is the exponential decay constant, g_{sat} is the saturation conductance value and t_i are the spike times. $p_{\text{rise}}(t)$ is triggered for every presynaptic spike that the neuron receives. Longer t_{rise} will result in longer exponential rise and thus a higher peak. Minimizing t_{rise} avoids saturation and matches $h(t) = \tau^{-1}e^{-t/\tau}$ more closely.

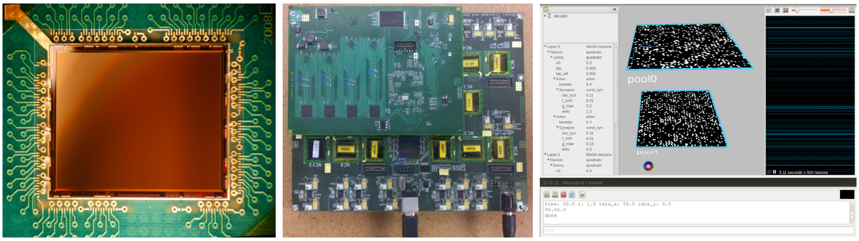
These synapses' programmable reversal potential (e_{rev}) supports positive and negative weights. An input spike causes the synapse to drive the membrane potential to e_{rev}. If e_{rev} is greater than the membrane's resting potential, the membrane depolarizes, realizing a positive weight (excitatory synapse). If e_{rev} is lower than the membrane's resting potential, the membrane hyperpolarizes realizing a negative weight (inhibitory synapse). These two effects are not necessarily balanced because their driving forces change in opposite ways with the membrane potential. This imbalance is minimized by setting e_{rev} much higher or much lower than v, such that their difference is much larger than v's changes.

Programmable interconnection weights are implemented probabilistically. In NEF, w_{ij} specifies the amplitude of the synaptic current (i.e., g_{sat}) that is fed as input to postsynaptic neuron j when presynaptic neuron i spikes. In our probabilistic implementation, the input strength does not vary. Instead, w_{ij} specifies the fraction of presynaptic neuron i's spikes that reach postsynaptic neuron j [8]. This approach eliminates the need for a digital-to-analog converter (assuming that the weights are stored digitally) and an analog multiplier. It also reduces bandwidth requirements, since the probabilities are usually very low (small weights).

The ability to utilize the fabrication process' imperfections (i.e., transistor mismatch) is one of NEF's attractions. The silicon neuron's properties (threshold voltage and leakage conductance) are determined by its transistors' physical properties (width, length and doping). Variations in these properties cause normalization factors to vary from one neuron to another, resulting in a distribution of bias and gain parameter values [5]. As a result, identically designed neurons have different conductance-to-spike-rate functions, and hence we do not need to set different bias currents and gains for each neuron. However, mismatch in the silicon synapse's properties (g_{sat}, e_{rev}, t_{rise} τ_{syn}) is detrimental because it imbalances excitation and inhibition, as the same circuit is used for all the synapses (of a given type) a particular neuron receives. We mitigated this by modifying the algorithm that finds the decoding weights (see Section 5).

4 System Description

We use Neurogrid, a neumorphic system with one million spiking neurons, for analog computation and a daughterboard, with an FPGA and a bank of SRAMs,



```
Layer 0            64x64 neurons
  Neuron           quadratic
    Soma           quadratic
      x0           0.3
      tau          0.003
      tau_ref      0.001
    Arbor          arbor
      lambda       0.4
      Synapse      cond_syn
        tau_syn    0.11
        t_xmt      0.01
        g_max      0.2
        erev       1.3
    Arbor          arbor
      lambda       0.4
      Synapse      cond_syn
        tau_syn    0.11
        t_xmt      0.01
        g_max      0.13
        erev       0.3
Layer 1            64x64 neurons
  Neuron           quadratic
    Soma           quadratic
      x0           0.3
```

```
time: 20.0 r: 1.0 rate_e: 50.0 rate_i: 0.0
50.00.0
done
>>>
```

Fig. 2. Neurogrid. Spiking neural networks with up to 16 layers, each with up to 256×256 neurons, are modeled with sixteen 12×14 sq-mm Neurocores (*left*) assembled on a 6.5×7.5 sq-in circuit board (*center*, main board). Neurocores, connected in a binary tree with 91M spikes/s links (peak rate), relay their spikes to a routing table (*center*, daughterboard) that supports programmable weighted connections. Spike activity is visualized in real time (*right*). The entire 1M-neuron system consumes 3.1W.

for digital communication (Figure 2). Layers are mapped onto Neurogrid's 180 nm CMOS chips (Neurocores) and connected by relaying packets.

The daughterboard implements our probabilistic connection scheme. The SRAMs (*Cypress Semiconductor* CY7C1071DV33) store 32 MBytes with a 50 ns access time. The FPGA (*Xilinx* Spartan-3E, 100MHz clock) parses incoming packets to identify the source neuron's chip, row, and column address. This information is used to calculate a base address from which to begin reading SRAM entries. Each 32-bit entry specifies four weights (6-bit value, a sign bit and a control bit). A Bernoulli trial is performed by comparing a 7-bit random number with the 6-bit value, padded with one zero to yield a maximum probability of 0.5. If the random number is less than the weight, a packet is output that specifies the route to the corresponding target neuron's Neurocore as well as its row and column address. The synapse type is also included—excitatory or inhibitory—determined by the weight's sign. If the random number is more than the weight, the connection is ignored and the next entry is read.

With all-to-all connectivity, the FPGA can support $N_{\mathrm{L}} = 2175$ neurons per layer firing at an peak mean rate of $f_{\mathrm{avg}} = 16.9$ spikes/s, a total of $N_{\mathrm{L}}^2 f_{\mathrm{avg}} = 80\mathrm{M}$ connections per second. Four weights are delivered every 50 ns, and thus each connection is processed in $t_{\mathrm{wgt}} = 12.5$ ns. Hence, N_{L} cannot exceed $1/\sqrt{f_{\mathrm{avg}} t_{\mathrm{wgt}}} = 2175$. Since the average weight is less than 1/16 in practice, no more than 5M packets per second are sent to Neurogrid. Thus, the 200 ns it takes to output a packet—four clock cycles per word—does not limit performance.

Larger networks can be supported by exploiting Neurogrid's local arbor mechanism, which efficiently distributes synaptic input by relaying analog signals between neighboring neurons [9]. A neuron r neurons away from the arbor's center receives a synaptic input that decays as $\lambda^r/\sqrt{r}$ (λ is programmable for each synapse type). This method enables us to increase connections by a factor of d^2, where d is the arbor's diameter, defined as $2/\ln \lambda$—twice the distance at which the current decays by a factor of e. It also produces distributions of bias currents

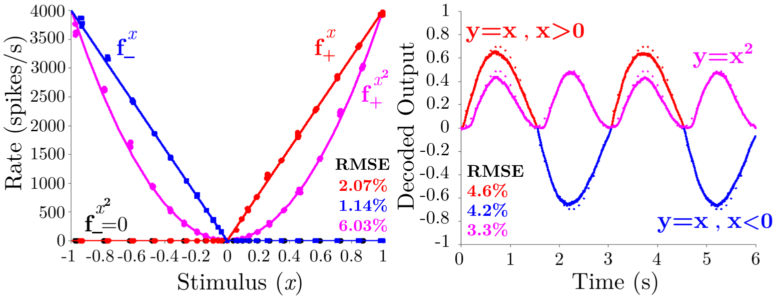


Fig. 3. Representation and Transformation. First (*left*) and second (*right*) layers accurately represent and transform static (*left*) and time-varying (*right*) values of $f_\pm = \max(\pm x, 0)$ and $\max(\pm x^2, 0)$, decoded with 0.1 s time-bins. $N_\mathrm{L} = 4\mathrm{K}$, $m = 50$ (an additional 50 distinct stimulus values were used for testing), $f_\mathrm{max} = 4000$ spikes/s, $\tau_\mathrm{m} = 6$ ms, $t_\mathrm{ref} = 1$ ms, $\tau_\mathrm{syn} = 0.1$ s, $t_\mathrm{rise} = 0.2$ ms, $g_\mathrm{sat} = 600$, $e_\mathrm{rev} = 4$ or 0.2, $\lambda = 0.45$ or 0.5, and $d = 8$. The linear and quadratic network's 1356 and 2927 neurons with non-zero weights fired 3.56 and 2.75 spikes/s/neuron, respectively, on average.

and gains larger than those provided by transistor mismatch, which is desirable for NEF. Using arbors yields $N_\mathrm{L} = d/\sqrt{f_\mathrm{avg} t_\mathrm{wgt}} = 17400$ for $d = 8$. This structured synaptic organization, whereby neighboring neurons receive similar inputs, is akin to the cortex's columnar organization.

5 Implementation Results

We demonstrated NEF's three principles in Neurogrid by communicating a scalar stimulus, quadratically transforming its value, and computing its time integral. We deliver the scalar stimulus ($-1 \le x \le +1$) in the form of Poisson spike trains ($f_\pm$) to the excitatory and inhibitory synapses.

To represent the scalar x, encoding vectors, $\tilde{\phi}_i = \pm 1$, are randomly assigned to the N_L/d^2 arbor centers where $f_\pm = f_\mathrm{max} \max(\pm x, 0)$ are applied. For $x = -1$, neurons with $\tilde{\phi}_i = +1$ receive 0 and f_max spikes/s at their excitatory and inhibitory synapses, respectively. For $x = +1$, the synapses receive f_max and 0 spikes/s, respectively. For $x = 0$, both synapses receive 0 spikes/s. The reverse is true for $\tilde{\phi}_i = -1$.

We compute decoders to recreate $f_\pm$ using convex optimization (`CVX` package in MATLAB), constraining the decoding weights to be non-negative (the next paragraph explains why) and to not exceed the FPGA's maximum probability. The N_L decoding weights, $\mathbf{\Phi}^{\mathbf{f}_\pm} = [\phi_1^{\mathbf{f}_\pm}, \ldots, \phi_{N_\mathrm{L}}^{\mathbf{f}_\pm}]^T$, satisfy:

$$\text{minimize } ||\mathbf{M}\mathbf{\Phi}^{\mathbf{f}_\pm} - \mathbf{f}_\pm||_2 \ \text{ subject to } \ 0 \le \phi_i^{\mathbf{f}_\pm} \le 0.5 \qquad (6)$$

where $\mathbf{M}$ is a $m \times N_\mathrm{L}$ matrix of measured spike rates, with N_L columns corresponding to the neurons and m rows corresponding to the applied spike input

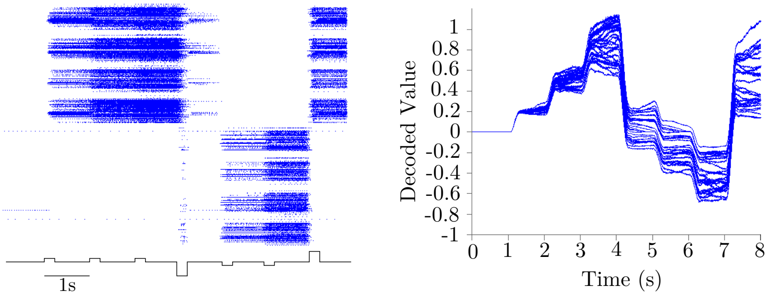


Fig. 4. Dynamics. Left: Spike rasters of 1237 neurons that had non-zero recurrent weights (*top*) in response to a train of 0.2 s long stimuli (*bottom*) presented every second with amplitudes of +1, +1, +1, −4, −1, −1 and +3. The banded structure results from local arbors. Right: Decoded integral of the stimuli (30 trials) reveal random walks that converge and diverge near particular values—evidence of local stable and unstable points. Parameters same as in Figure 3.

rates, which are specified by $\mathbf{f}_{\pm}$, a $m \times 1$ vector. We build decoders to obtain $f_{\pm}^{x^2} = f_{\max} \max(\pm x^2, 0)$ using the same approach (Figure 3, *left*).

To communicate the scalar ($y = x$) or transform it quadratically ($y = x^2$), we use the previously computed decoding weights to program the weight that connects source layer neuron i to target layer arbor center j. Specifically, we program the excitatory and inhibitory weights to $w_{ij}^{\pm} = \max(\langle\tilde{\phi}_j \cdot \phi_i^{\mathbf{f}_{\pm}}\rangle, 0) - \min(\langle\tilde{\phi}_j \cdot \phi_i^{\mathbf{f}_{\mp}}\rangle, 0)$, such that $w_{ij} = w_{ij}^{+} - w_{ij}^{-}$. This formulation delivers an estimate of $f_{\pm}$ to the target layer, except that it will use the opposite synapse type if $\phi_i^{\mathbf{f}_{\pm}} < 0$. For instance, if $f_{\pm}$ both have 10 negatively weighted spikes per second, then instead of receiving 40 ($= 50 - 10$) and 10 ($= 20 - 10$) spikes/s, the excitatory and inhibitory synapses would receive 60 and 30 spikes/s. If mismatch (see Section 3) makes the excitatory synapse stronger, these additional spikes will have a net excitatory effect. This bias would make the target neuron spike at a higher rate, which would lead to a decoding error. Non-negative decoding weights avoid this error and yields accurate communication and transformation (Figure 3, *right*).

To integrate the scalar ($\dot{x} = u$), we use the decoding weights for $f_{\pm} = f_{\max} \max(\pm x, 0)$ to program the weight w_{ij} that connects neuron i to arbor center j within the same layer ($\mathbf{A}' = \mathbf{I}$) and apply input spike rates scaled by τ_{syn} ($\mathbf{B}' = \tau_{\text{syn}}\mathbf{I}$). At 2 s, the integrator's time constant is over an order of magnitude greater than $\tau_{\text{syn}} = 0.1$ s (Figure 4). Over multiple trials, the mean and variance of the integrator's value change with a time constant of 2.0 s (drift) and a rate of 4.9×10^{-4}/s (diffusion), respectively. These values can be reduced by increasing $f_{\max}$ or τ_{syn}, as each neuron estimates $f_{\pm}$ using the $\tau_{\text{syn}} f_{\pm}$ spikes it received in the last τ_{syn} seconds. The signal-to-noise ratio should improve as $\sqrt{\tau_{\text{syn}} f_{\max}}$ (assuming Poisson statistics). However, the hardware's minimum t_{rise} (0.2 ms) limited $f_{\max}$ to 4000 spikes/s, compared to $80 \times 4096 = 320$K for NEF.

6 Conclusion

We have demonstrated NEF's three principles on neuromorphic hardware, successfully performing arbitrary mathematical computations using silicon neurons. Unlike NEF's deterministic weights, we used probabilistic weights and still achieved accurate results. Our results readily extend to multiple dimensions, thus supporting any control-theoretic algorithm. One example is a Kalman filter-based decoder for brain-machine interfaces [10], and we can now envision a low-power, fully implantable neuromorphic chip to implement it.

Acknowledgements. We thank Emmett McQuinn for supporting Neurogrid's user interface, Dan Neil for supporting Neurogrid's hardware, and Kimberly Chin for administrative assistance. This collaborative project started at the 2011 Telluride Neuromorphic Engineering Workshop and was supported by a NIH Director's Pioneer Award (DPI-OD000965, K. Boahen), a NIH/NINDS Transformative Research Award (R01NS076460, K. Boahen), and a NSF Graduate Research Fellowship (S. Fok).

References

1. Sarpeshkar, R., Delbruck, T., Mead, C.A.: White noise in MOS transistors and resistors. IEEE Circuits and Devices Magazine 9(6), 23–29 (1993)
2. Eliasmith, C., Anderson, C.H.: Neural engineering: computation, representation, and dynamics in neurobiological systems. MIT Press, Cambridge (2003)
3. Boahen, K.: A Burst-Mode Word-Serial Address-Event Link-I: Transmitter Design. IEEE Transactions on Circuits and Systems I 51(7), 1269–1280 (2004)
4. Silver, R., Boahen, K., Grillner, S., Kopell, N., Olsen, K.L.: Neurotech for neuroscience: unifying concepts, organizing principles, and emerging tools. Journal of Neuroscience 27(44), 11807–11819 (2007)
5. Gao, P., Benjamin, B.V., Boahen, K.: Dynamical system guided mapping of quantitative neuronal models onto neuromorphic hardware. IEEE Transactions on Circuits and Systems (in press, 2012)
6. Benjamin, B.V., Arthur, J.V., Gao, P., Merolla, P., Boahen, K.: A Superposable Silicon Synapse with Programmable Reversal Potential. In: International Conference of the IEEE Engineering and Medicine in Biology Society (in press, 2012)
7. Arthur, J.V., Boahen, K.A.: Synchrony in Silicon: The Gamma Rhythm. IEEE Transactions on Neural Networks 18(6), 1815–1825 (2007)
8. Goldberg, D.H., Cauwenberghs, G., Andreou, A.G.: Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons. Neural Netw. 14(6-7), 781–793 (2001)
9. Andreou, A.G., Boahen, K.: Translinear circuits in subthreshold MOS. J. Anal. Integr. Circuits Signal Process 9, 141–166 (1996)
10. Dethier, J., Nuyujukian, P., Eliasmith, C., Stewart, T., Elassaad, S.A., Shenoy, K.V., Boahen, K.: A Brain-Machine Interface Operating with a Real-Time Spiking Neural Network Control Algorithm. In: Advances in Neural Information Processing Systems, vol. 24 (2011)