Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations

This paper describes the design of the first hardware system to provide computational neuroscientists with the capability of performing biological real-time simulations of a million neurons and their synaptic connections.

By Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean-Marie Bussat, *Member IEEE*, Rodrigo Alvarez-Icaza, John V. Arthur, Paul A. Merolla, and Kwabena Boahen, *Senior Member IEEE*

ABSTRACT | In this paper, we describe the design of Neurogrid, a neuromorphic system for simulating large-scale neural models in real time. Neuromorphic systems realize the function of biological neural systems by emulating their structure. Designers of such systems face three major design choices: 1) whether to emulate the four neural elements—axonal arbor, synapse, dendritic tree, and soma—with dedicated or shared electronic circuits; 2) whether to implement these electronic circuits in an analog or digital manner; and 3) whether to interconnect arrays of these silicon neurons with a mesh or a tree network. The choices we made were: 1) we emulated all neural elements except the soma with shared electronic circuits; this choice maximized the number of synaptic connections; 2) we realized all electronic circuits except those for axonal arbors in an analog manner; this choice maximized energy efficiency; and 3) we interconnected neural arrays in a tree network; this choice maximized throughput. These three choices made it possible to simulate a million neurons with billions of synaptic connections in real time—for the first time—using 16 *Neurocores* integrated on a board that consumes three watts.

KEYWORDS | Analog circuits; application specific integrated circuits; asynchronous circuits; brain modeling; computational neuroscience; interconnection networks; mixed analog-digital integrated circuits; neural network hardware; neuromorphic electronic systems

I. SIMULATING LARGE-SCALE NEURAL MODELS

Large-scale neural models seek to integrate experimental findings across multiple levels of investigation in order to explain how intelligent behavior arises from bioelectrical processes at spatial and temporal scales six orders of magnitude smaller (from nanometers to millimeters and from microseconds to seconds). Due to prohibitively expensive computing costs, very few models bridge this gap, failing to make behaviorally relevant predictions [1]. A personal computer simulates a mouse-scale cortex model (2.5×10^6 neurons) 9000 times slower than a real mouse brain operates [2], while using 40 000 times more power (400 W versus 10 mW [3]). Simulating a human-scale cortex model (2×10^{10} neurons), the Human Brain Project's goal, is

Manuscript received November 27, 2013; revised February 9, 2014; accepted February 25, 2014. Date of publication April 24, 2014; date of current version April 28, 2014. This work was supported by the National Institutes of Health Director's Pioneer Award DPI-OD000965 and Transformative Research Award R01NS076460.

B. V. Benjamin is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA.
P. Gao and **K. Boahen** are with the Department of Bioengineering, Stanford University, Stanford, CA 94305 USA (e-mail: boahen@stanford.edu).
E. McQuinn is with In-Depth, Inc., San Francisco, CA 94107 USA.
S. Choudhary is with Intel Corporation, Santa Clara, CA 95054-1537 USA.
A. R. Chandrasekaran is with Mad Street Den, Bangalore 782003, India.
J.-M. Bussat was with the Department of Bioengineering, Stanford University, Stanford, CA 94305 USA.
R. Alvarez-Icaza, **J. V. Arthur**, and **P. A. Merolla** are with the Almaden Research Center, IBM, San Jose, CA 95120 USA.

Digital Object Identifier: 10.1109/JPROC.2014.2313565

projected to require an exascale supercomputer (10^{18} flops) [4] and as much power as a quarter-million households (0.5 GW) [5]. Hence, large-scale neural modeling's potential has hardly been tapped.

Several groups are developing custom computing platforms with the stated aim of simulating large-scale neural models affordably. The University of Manchester SpiNNaker project aims to improve the performance of software simulations by integrating 18 mobile processors onto a single die [6], [7]. The IBM SyNAPSE project (GoldenGate chip) aims to overcome the memory bottleneck such simulations face by replacing virtualization with custom-designed digital electronic circuits that are each dedicated to emulating a single neural element [8]. The Heidelberg University BrainScales project (HICANN chip) aims to reduce the number of transistors these electronic circuits require by using an analog approach [9]. Our Neurogrid project (Fig. 1), which also uses an analog approach, aims to reduce transistor count further by sharing synapse and dendritic tree circuits [10]. Thus, these four projects have adopted radically different architectures. They also use different interconnection networks to route spikes between arrays of neural elements.

In this paper, we present an analysis of various neuromorphic architectures' and spike-routing networks' scaling properties, which informed Neurogrid's design choices (Section II); describe the complete Neurogrid system (Section III); provide detailed descriptions of Neurogrid's neuron circuit (Section IV) and chip design (Section V); dissect Neurogrid's energy consumption (Section VI); compare its area, energy, and time per synapse or synaptic activation with the other systems under development (Section VII); and discuss insights gleaned from these comparisons (Section VIII).

II. NEUROMORPHIC ARCHITECTURES

Neuromorphic hardware [11], usually realized with axonal arbor, synapse, dendritic tree, and soma elements, may be

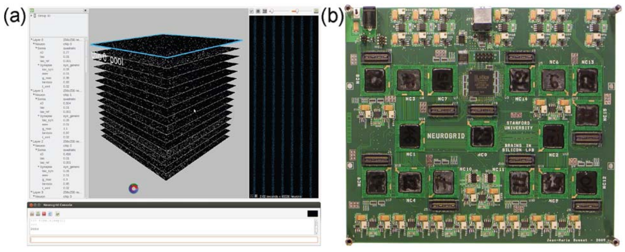


Fig. 1. ***Neurogrid. (a) GUI: Enables a user to change his or her model parameters (left), view spike activity in the model's various layers (middle), plot spike rasters from a selected neural layer (right), and enter commands (bottom). (b) Board: Each neural layer is simulated by up to 256 × 256 silicon neurons on each of 16 Neurocores integrated on a 6.5 × 7.5 in^2 board.***

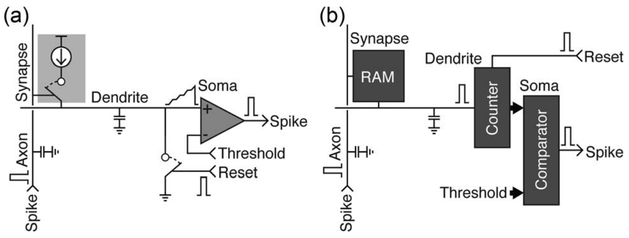


Fig. 2. ***Analog and digital silicon neurons. (a) Analog implementation: Incoming spikes on the vertical wire (axon) meter charge (synapse) onto the horizontal wire (dendrite), whose capacitance integrates the charge. The comparator (soma) compares the resulting voltage with a threshold and triggers an outgoing spike when the threshold is exceeded. The capacitor is then discharged (reset) and the cycle starts over. (b) Digital implementation: A counter is incremented (dendrite) each time a 1 is read out of a bit cell (synapse), triggered by the incoming spike (axon). The counter's output is compared (soma) with a digitally stored threshold and a spike is triggered when it is suprathreshold. The counter is then reset and the cycle starts over.***

categorized by architecture (i.e., whether elements are dedicated or shared) and implementation (i.e., whether elements are analog or digital). These distinctions yield many neuromorphic hardware realizations. In this section, we briefly review four neural-element array architectures and five hardware realizations, as well as two interconnection network topologies for routing spikes. In addition to bringing readers unfamiliar with the field up to speed, this review motivates the architectural and implementation choices we made in designing Neurogrid. Before proceeding, we describe simple analog and digital implementations of the four neural elements.

Axonal arbors, synapses, dendritic trees, and somas may be implemented in an analog or digital fashion: In the simplest fully analog implementation, these elements are emulated by a wire, a switched current-source, another wire, and a comparator, respectively [12] [Fig. 2(a)]. The switched current source's bias voltage—which determines the synaptic weight—is stored in an analog [13] or digital [14] manner; the latter requires a digital-to-analog converter. In the simplest fully digital implementation, the switched current source is replaced with a bit cell, the axon and dendrite function as word and bit lines, respectively, and integration and comparison are implemented digitally [8], [15] [Fig. 2(b)].

A. Four Architectures

We review four distinct architectures: fully dedicated (FD), shared axon (SA), shared synapse (SS), and shared dendrite (SD).[1] In FD, first realized in very large scale integration (VLSI) 29 years ago [17], [18], all elements are dedicated [Fig. 3(a)]. A fully connected network of N

[1]Shared-soma architectures have been reviewed extensively elsewhere [16].

neurons requires N^2 synapse elements—a feature this architecture shares with SA.

In SA, first realized 22 years ago using the address–event representation (AER) [19], [20], each neuron is assigned a unique address. Each time any neuron spikes, its address is encoded by a transmitter, communicated on a digital bus, and decoded by a receiver [Fig. 3(b)]. This time-multiplexing leverages digital's speed to reduce the number of wires from N to $\log_2(N)$ without any noticeable penalty as long as the bus does not become overly congested [21].

In SS, first realized 24 years ago [22], [23], only N electronic circuits are required to fully connect N neurons, rather than N^2, a feature shared with SD [Fig. 3(c)]. SS uses RAM to realize axonal branching—instead of connecting a dedicated wire to multiple synapse circuits as FD and SA do. In the original realization, which was all digital, each shared-synapse circuit retrieved the weight to be applied from a local RAM using the address on the AER bus (unlike in the figure) [22], [23]. As it limits an individual synapse's effect to its allotted time slot, this digital realization is said to be time-multiplexed. SS is attractive when connectivity is sparse—a situation in which FD and SA waste hardware implementing weights that are zero.

More recent shared-synapse designs have used an analog realization and consolidated all the RAM in a single monolithic block [24]–[26], which may be embedded in the chip [27]–[29]. The target neurons' addresses and weights are written to a location specified by the source neuron's address and retrieved sequentially. The analog realization allows an individual synapse's effect to extend beyond its allotted time slot, decaying exponentially with time. Because this behavior is realized via the principle of linear superposition, such shared-synapse circuits are said to be superposable [30]–[32], as opposed to time-multiplexed. A resistor–capacitor circuit (which may be implemented with transistors [33]) is used; it essentially performs temporal low-pass filtering.

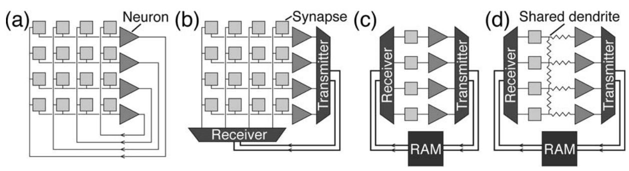
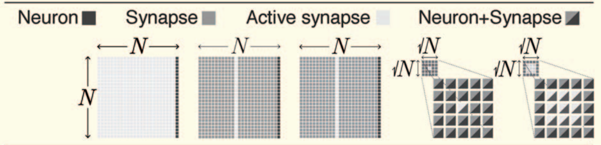


Fig. 3. ***Functionally equivalent architectures. (a) Fully dedicated: Hardware elements (for axonal arbors, synapses, dendritic trees, and somas) are dedicated to individual neural elements. Thus, there is a one-to-one correspondence between the chip's elements and the neural network's elements. (b) Shared axon: A common set of wires is shared by all of a neuronal population's axons. (c) Shared synapse: A single electronic circuit is shared by all of a neuron's synapses; a RAM is programmed to route all of its presynaptic spikes to this circuit. (d) Shared dendrite: A single resistive network is shared by a neuronal population's (overlapping) dendritic trees.***

Table 1 Architecture Comparison



Per	Dedicated	Shared Axon		Shared Synapse/Dendrite[†]	
Synapse	FDA	SAH	SAD	SSH	SDH[††]
A	1	1	1	$1/N$	$1/N$
E	2	2	$1+N$	$2\sqrt{N}$	2
T	$1/N$	1	2	$\sqrt{N}$	1
$A \cdot E \cdot T$	$2/N$	2	$2(1+N)$	2	$2/N$

[†] N synapse circuits are tiled in a $\sqrt{N} \times \sqrt{N}$ array, with the N neuron circuits interspersed.
[††] A delivered spike triggers synaptic input in $\sqrt{N}$ neighboring neurons.

In SD, first realized 10 years ago [34], [35], each shared-synapse circuit feeds its neuron's neighbors as well. This arrangement models a cluster of synapses formed by an axon onto dendrtitic branches from nearby neurons (i.e., overlapping dendritic trees). The shared-dendrite circuit makes this possible by applying the superposition principle in space instead of in time. In its analog implementation, a resistive network (which may also be implemented with transistors [36]) is used; it essentially performs spatial low-pass filtering.

B. Realization Comparison

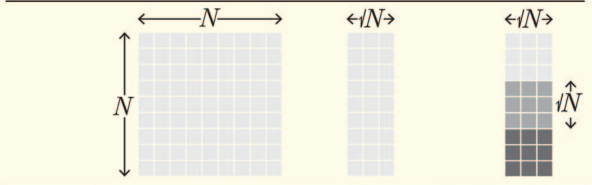
We compare a total of five realizations: fully dedicated analog (FDA), shared axon hybrid (SAH), shared axon digital (SAD), shared synapse hybrid (SSH), and shared dendrite hybrid (SDH). The hybrid realizations are all analog except for their axonal arbors. To compare these five realizations, we calculated how the area (A) a single synapse occupies, the energy (E) consumed when it is activated, and the time (T) it takes to do so scale with the number of neurons (N) in a fully connected network (Table 1). We ignore other important metrics, in particular precision. This is intentional because neuromorphic systems seek to compute with low-precision elements; their *raison d'être* is to achieve precision at the network level by leveraging collective computation.[2]

A is equal to A_{array}/N^2, where A_{array} is the array's area, which is N^2 units for FDA, SAH, and SAD, and N units for SSH and SDH.

E is equal to $(E_{\text{axon}} + n_{\text{axon}} E_{\text{dend}})/n_{\text{axon}}$, where E_{axon} and E_{dend} are, respectively, the energy required to activate an axon and a dendrite, obtained by multiplying the wire's capacitance (proportional to its length) by its voltage swing; n_{axon} is the number of synapses the axon contacts. For the axon, the voltage swing is 1 unit (normalized by the supply voltage VDD); hence, E_{axon} is N units for FDA, SAH,

[2]The relationship among precision, power, and area is well understood at the element level; it favors analog over digital at low precision (< 8 b [37]).

Table 2 RAM Costs



Per Synapse	SSH	SDH	Banked
A	1	$1/\sqrt{N}$	$1/\sqrt{N}$
E	$1+N$	$1/\sqrt{N}+\sqrt{N}$	$1/\sqrt{N}+1$
T	2	$1/\sqrt{N}+1$	$2/\sqrt{N}$
$A \cdot E \cdot T$	$2\,(1+N)$	≈ 1	$\approx 2/N$

and SAD, and $2\sqrt{N}$ units for SSH and SDH. For the dendrite, the voltage swing is 1 unit for digital (full swing) and $1/N$ units for analog ($1/N^{\mathrm{th}}$ full swing).[3] Hence, E_{dend} is N units for SAD, 1 unit for FDA and SAH, and $1/N$ units for SSH and SDH. n_{axon} is N for FDA, SAH, and SAD, 1 for SSH, and $\sqrt{N}$ for SDH.

T is equal to $t_{\mathrm{axon}}/(n_{\mathrm{par}} n_{\mathrm{axon}})$, where t_{axon} is the time it takes to activate an axon (proportional to its capacitance) and n_{par} is the number of axons that can be activated in parallel (1 in all cases but FDA, where it is N). Except for SAD, where we must add the time it takes to activate a dendrite ($t_{\mathrm{dend}} = N$ units) divided by the number of dendrites activated in parallel ($n_{\mathrm{axon}} = N$). t_{axon} is N units for FDA, SAH, and SAD and $\sqrt{N}$ units for SSH and SDH. Note that full connectivity requires $T = 1/N^2$ (i.e., N^2 bandwidth), which none of these realizations achieve.

We also have to account for the cost of SSH's and SDH's RAM, whose size equals the number of addressable locations times the number of words per location (Table 2). SSH requires an $N \times N$ RAM, whose A, E, and T scale like SAD's synapse array's. SDH only requires an $N \times \sqrt{N}$ RAM, as the shared dendrite provides an additional fanout of $\sqrt{N}$. As a result, A and E are $\sqrt{N}$ times smaller, and hence SDH's AET product scales like SAH's. Partitioning this RAM into $\sqrt{N}$ banks (of size $\sqrt{N} \times \sqrt{N}$) reduces E and T by an additional factor of $\sqrt{N}$, making SDH's AET product scale like FDA's.

When the cost metric is the product of A, E, and T, FDA and SDH tie for the lowest cost and SAD has the highest, N^2 times more costly. Giving A, E, and T the same exponents favors achieving performance through parallelism rather than by burning power.[4] Interestingly, FDA achieves its cost-effectiveness by minimizing T, while SDH does it by minimizing A (see Table 1). That is, for an N-neuron network, the former runs N times faster but the latter uses N times less area—and constrains connectivity patterns. Thus, FDA is the best choice for applications that run faster than real time and have arbitrary connectivity, such as modeling neural development (HICANN's goal [9]), while SDH is the best choice for applications that require lots of neurons but have mostly local connectivity, such as modeling neocortex (Neurogrid's goal [10]).

Our conclusions do not change when static power dissipation (due to bias or leakage currents) is included in the model. FDA and SDH use complementary strategies to reduce static dissipation. FDA runs its N^2 physical synapses N times faster, reflected in its N-fold lower T, decreasing static energy proportionally. SDH realizes its N^2 synapses with N times fewer transistors, reflected in its N-fold lower A, decreasing static power proportionally. Since both have $AT = 1$, static energy makes similar contributions to E in both. This result predicts that FDA and SDH have similar static power (per synapse) and suggests how to extend the model to include it.

C. Spike-Routing Networks

Meshes [6], [39] and trees [40]–[42] have been explored for routing spikes between neural-element arrays (called nodes), with a route appended to the neuron's address to create a packet. We compare how these networks' throughput and latency scale with the number of nodes n.[5] Because each neuron is connected to thousands of others, high bandwidth is required. And because spike times are used to encode information, extremely short latency is required.

Meshes offer high bandwidth due to their large channel bisection ($\sqrt{n}$ for n nodes), but have long latency due to their large diameter ($\sqrt{n}$).[6] Trees offer short latency (diameter is $\log n$) but have low bandwidth (channel bisection is 1). Unlike meshes, however, trees support deadlock-free multicast communication (i.e., routing one packet to many destinations), enabling them to utilize their limited bandwidth efficiently, and thereby maximize throughput.

Deadlock occurs when a packet is waiting for a packet ahead of it to move, which in turn is waiting for a packet ahead of it, and so on, and these dependencies form a closed cycle. In this case, none of the packets make progress toward their destinations. Hence, the routing network is said to be deadlocked. For unicast communication (i.e., one-to-one routing), meshes are provably deadlock-free when dimension-order routing or virtual channels are used [43]. However, this solution does not work for multicast routing, which introduces additional

[3]It makes the neuron's spike rate similar to the N neurons synapsing onto it.

[4]T can be halved by giving each neuron two shared-synapse circuits that can be activated concurrently, which doubles A; or by doubling the voltage, which quadruples E [38].

[5]We ignore all terms in the expressions except the one with the largest exponent, and we drop this term's coefficient (e.g., $(3/2)n^2 + 8n \rightarrow n^2$).

[6]Channel bisection is defined as the minimum number of links connecting two halves of the network across all possible bisections [43]. Diameter is defined as the longest minimal path, in number of links transversed, between any pair of nodes [43].

Table 3 Mesh Versus Tree for All-to-All Traffic

Packets	Mesh	Tree	
relayed	Unicast	Unicast	Multicast
Average	$n^{3/2}$	$n\log(n)$	n
Peak	$n^{3/2}$	n^2	n

packet dependencies [44]. Trees are provably deadlock-free in the unicast case when up-down routing is used [45]–[47]. This solution works for the multicast case as well if branching (copying a packet from a parent to its two children) is restricted to the downward phase [42].

The number of packets each node in a mesh or a tree relays can be obtained analytically for all-to-all traffic (Table 3) [42], [47]. In this extreme benchmark, n nodes exchange a total of n^2 packets. For the mesh, which is restricted to unicast to avoid deadlock, each node relays $n^{3/2}$ packets (traffic is uniformly distributed). For the tree, the root relays n^2 packets for unicast and n for multicast. Thus, multicast cuts the root's traffic by a factor of n (equal to the fanout), relieving this bottleneck. As a result, the tree's peak node traffic is $\sqrt{n}$ less than the mesh's, and its latency is $\sqrt{n}/\log(n)$ times shorter.

To summarize, in addition to its lower latency, the tree offers higher throughput than the mesh if the application can utilize the former's multicast capability, and it requires roughly a third less resources than the mesh does.[7] As the neocortical simulations Neurogrid targets can use multicast to realize secondary axon-branching, we chose the tree.

The routing network may be incorporated into the scaling model developed in Section II-B by including its area in calculating A, its energy in calculating E, and its bandwidth in calculating T (Vainbrand and Ginosar [48] analyze how these quantities scale for various topologies). We illustrate this for Neurogrid in Sections VI and VII.

III. NEUROGRID

Neurogrid has two main components: software to perform interactive visualization and hardware to perform real-time simulation (Fig. 4). Neurogrid's software stack is composed of a user interface (UI), a hardware abstraction layer (HAL), and driver components (Driver). UI allows a user to specify models of neural networks to be simulated, interact with the simulations, and visualize the results in real time. HAL maps the parsed model description to Neurogrid's electronic circuits. Driver programs this mapping on to Neurocores over USB using Neurogrid packets.

[7]The hardware grows quadratically with the number of connections a node has, which, including the connection to its array, is four for a binary tree and five for a 2-D mesh.

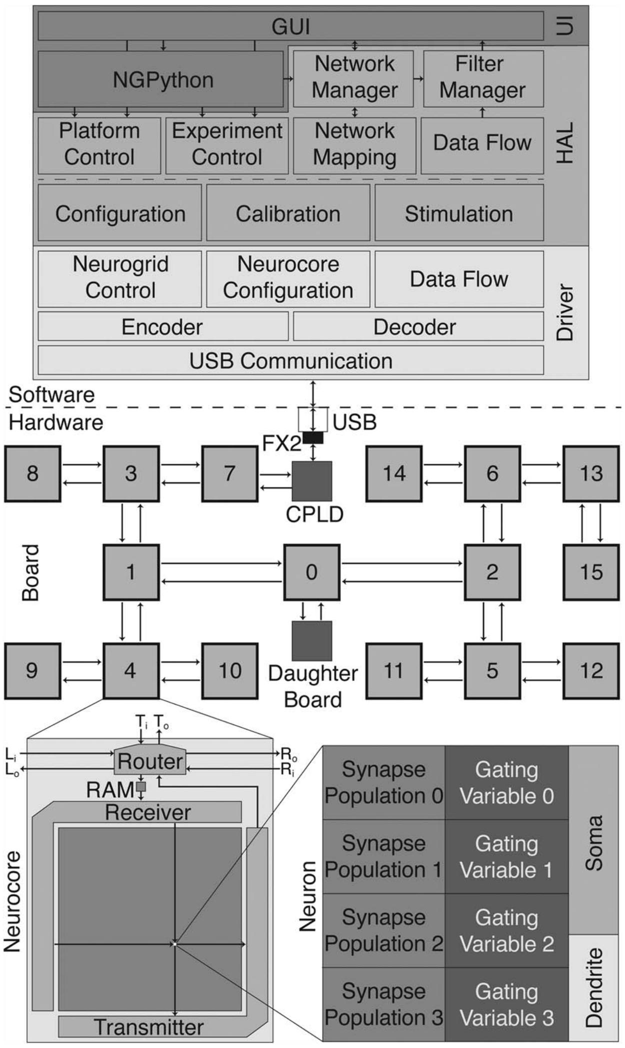


Fig. 4. ***Neurogrid's software and hardware. UI: NGPython allows a user to specify neuronal models in the Python programming environment; GUI provides an interface to control simulations as well as to visualize the results in real time. HAL: Network and Filter Manager provide the GUI with the simulated network's connectivity and activity, respectively; Platform Control converts the user's neural models' parameters to bias currents for Neurogrid's electronic circuits; Experiment Control starts, stops, and resets both simulation and stimulation; Network Mapping converts the models' connectivity to router configurations; Data Flow translates data from model space to hardware space and vice versa. Driver: Neurogrid Control handles global resets and bring-up; Neurocore Configuration programs bias currents and router configurations; Data Flow creates Neurogrid packets; Encoder converts Neurogrid packets to USB format and Decoder converts USB data to Neurogrid packets. Board: FX2 handles USB communication with the host; CPLD converts USB data to Neurogrid packets and vice versa, as well as intersperses time stamps with outgoing data (host bound); Daughterboard realizes primary axonal branching. Neurocore: Router communicates packets with the Neurocore's parent (through T_i/T_o) and two children (through L_i/L_o and R_i/R_o); RAM supports reconfigurable connectivity (a second RAM supports programmable biases); Receiver delivers spikes to silicon neuron array; Transmitter dispatches spikes from the array. Neuron: Consists of a soma, a dendrite, four gating variable and four synapse-population circuits.***

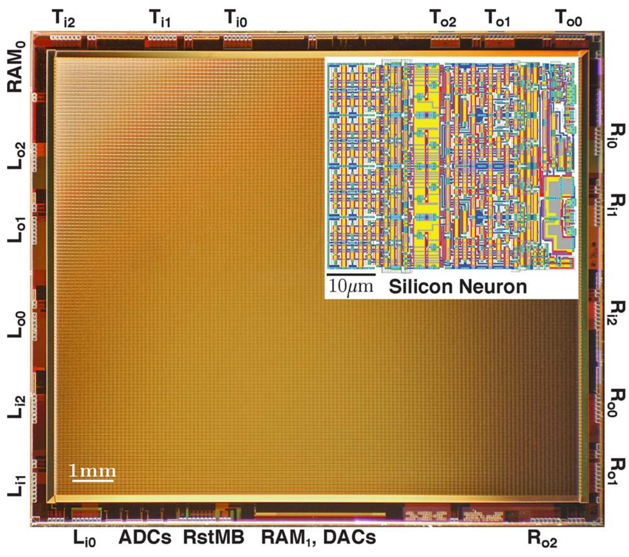


Fig. 5. ***Neurocore. RAM_0 provides 256 locations for target synapse types (or no connection). RAM_1 stores 18 configuration bits and 61 analog biases, common to all the Neurocore's silicon neurons. DACs produce the analog biases. RstMB provides five resets and generates DACs' reference current. ADCs digitize four analog signals from a selected neuron. T_{i0-2}, T_{o0-2}, L_{i0-2}, L_{o0-2}, R_{i0-2}, and R_{o0-2} communicate with parent or either child. The 12 $\times$ 14 mm^2 die, with 23 M transistors and 180 pads, was fabricated in a 180-nm complementary metal–oxide–semiconductor (CMOS) process. Insert: Layout of silicon neuron; it has 337 transistors (see Fig. 4 for subcircuit placement).***

A Neurogrid packet, used in Neurocore-to-Neurocore communication, is a sequence of 12-b words that specify a route, an address, an arbitrarily long payload, and a tailword, in that order. The route instructs a Neurocore to forward the packet to the next hop or consume it. The payload stipulates: Spike locations in a row of a silicon-neuron array, in which case the address identifies a Neurocore; data to be written to RAM, in which case the address specifies a location; or a sampled analog signal, in which case the address identifies an ADC (a Neurocore has four). The tailword signifies the packet's end.

The hardware consists of an Cypress EZ-USB FX2LP, a Lattice ispMACH CPLD, a daughterboard, and 16 Neurocores connected in a binary tree. The FX2 handles USB communications. The CPLD interfaces between the FX2 and the Neurocores. The daughterboard realizes primary axon-branching using a Xilinx Spartan-3E FPGA and eight Cypress 4MB SRAMs. A Neurocore (Fig. 5) has a 256 $\times$ 256 silicon-neuron array, a transmitter, a receiver, a router, and two RAMs. A neuron has a soma, a dendrite, four gating-variable and four synapse-population (i.e., shared synapse and dendrite) circuits. We describe these electronic circuits and the models they implement in Section IV. The transmitter, the receiver, and the router are described in Section V.

IV. NEURON

Neurogrid neuron's soma, dendrite, synapse-population and ion-channel-population circuits realize the dimensionless form of common biological neuronal models with MOS devices. Dimensionless models have fewer free parameters and can be realized on a wide variety of hardware platforms. However, MOS devices offer the highest integration density.

A. Dimensionless Models

Before describing the dimensionless models Neurogrid realizes (Fig. 6), we illustrate how models composed of conductors, capacitors, voltage, and current sources may be converted to dimensionless form.

Consider a passive membrane model with a capacitor C, a conductor G_{leak} with reversal potential E_{leak}, and a current source I_{in}. This circuit is described by

$$C\dot{V} = -G_{\text{leak}}(V - E_{\text{leak}}) + I_{\text{in}}$$

where V is the voltage across C. This equation has four parameters: C, G_{leak}, E_{leak}, and I_{in}, even though the model has only two degrees of freedom. To see this, change the reference voltage to E_{leak} and normalize with $G_{\text{leak}}V_{\text{n}}$ to give

$$\tau\dot{v} = -v + u \qquad (1)$$

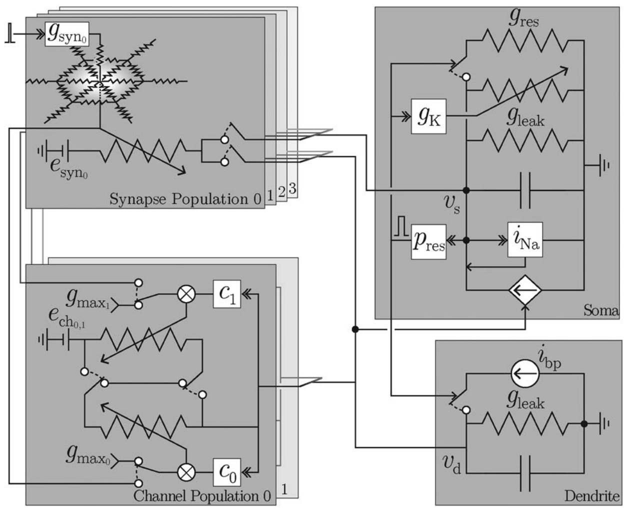


Fig. 6. ***Neurogrid's neuron model. Synapse Populations 0, 1, 2, and 3 generate a time-varying conductance ($g_{\text{syn}_{0-3}}$) in response to an input spike and drive neighboring neurons' soma or dendrite through a shared dendritic tree. Channel Populations 0 and 1 provide a pair of conductances that are dynamically activated or inactivated (c_{0-3}) by dendritic potential; their maximum conductance may be determined by Synapse Population. These conductances may be connected in series or in parallel to drive Dendrite. Dendrite also receives backpropagating spikes (i_{bp}) and drives Soma. Soma generates spikes, using a regenerative sodium current (i_{Na}), which triggers a reset pulse (p_{res}). A potassium conductance (g_{K}) activated at reset delays generation of the next spike.***

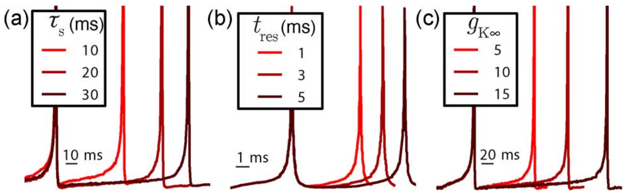


Fig. 7. ***Soma circuit's membrane traces. (a) Increasing τ_s: It increases the interspike interval by slowing integration. (b) Increasing t_{res}: It increases the interspike interval by resetting the membrane longer. (c) Increasing $g_{K\infty}$: It increases the interspike interval by producing larger increments in the potassium conductance.***

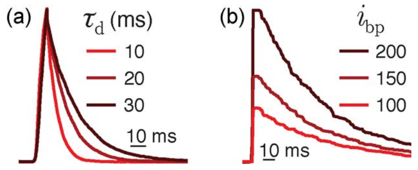


Fig. 8. ***Dendrite circuit's membrane traces. (a) Increasing τ_d: It increases decay time by slowing integration (traces were normalized with their peak values). (b) Increasing i_{bp}: It increases the current injected by each backpropagating spike.***

where $\tau = C/G_{\rm leak}$, $v = (V - E_{\rm leak})/V_{\rm n}$, and $u = I_{\rm in}/(G_{\rm leak}V_{\rm n})$. Thus, v is the voltage in units of $V_{\rm n}$ and u is the current in units of $G_{\rm leak}V_{\rm n}$. This equation has two parameters: τ and u, to match the model's two degrees of freedom. This approach is general: Any electrical model of a membrane can be made dimensionless by changing the reference voltage to $E_{\rm leak}$ and normalizing voltages with $V_{\rm n}$, conductances with $G_{\rm leak}$ and currents with $G_{\rm leak}V_{\rm n}$. Henceforth, we denote dimensionless equivalents of voltages with v, conductances with g, and currents with i.

Soma: The soma's dimensionless model is given by

$$\tau_s\dot{v}_s = -v_s + i_{\rm sin} + \tfrac{1}{2}v_s^2 - g_K v_s - g_{\rm res}v_s p_{\rm res}(t) + v_d \qquad (2)$$

where τ_s is the membrane time constant, $i_{\rm sin}$ is the input current, and v_d is the dendritic input (see Fig. 6 Soma). The quadratic positive feedback $v_s^2/2$ models the spike-generating sodium current [49]; the reset conductance $g_{\rm res}$, active for the duration $t_{\rm res}$ of a unit-amplitude pulse $p_{\rm res}$, models the refractory period; and the high-threshold potassium conductance g_K models spike-frequency adaptation. g_K is given by

$$\tau_K\dot{g}_K = -g_K + g_{K\infty}p_{\rm res}(t) \qquad (3)$$

where τ_K is the decay time constant and $g_{K\infty}$ is the saturation value. The soma may also receive synaptic inputs [see (6)]. Hardware realization of the soma model behaves as expected (Fig. 7).

Dendrite: The dendrite's dimensionless model is given by

$$\tau_d\dot{v}_d = -v_d + i_{\rm din} + i_{\rm bp}p_{\rm res}(t) + g_{\rm ch}(e_{\rm ch} - v_d) \qquad (4)$$

where τ_d is the membrane time constant, $i_{\rm din}$ is the input current, $i_{\rm bp}$ is the backpropagating input, and $g_{\rm ch}$ is the channel population's conductance, with reversal potential $e_{\rm ch}$ (see Fig. 6 Dendrite). The dendrite may also receive synaptic inputs [see (6)]. Hardware realization of the dendrite model behaves as expected (Fig. 8).

Synapse Population: The synapse population's dimensionless model is given by

$$\tau_{\rm syn}\dot{g}_{\rm syn} = -g_{\rm syn} + g_{\rm sat}p_{\rm rise}(t) \qquad (5)$$

where $\tau_{\rm syn}$ is the synaptic time constant and $g_{\rm sat}$ is the saturation conductance for the population (see Fig. 6 Synapse Population). The unit-amplitude pulse $p_{\rm rise}(t)$ is triggered by an input spike; its width $t_{\rm rise}$ models the duration for which neurotransmitter is available in the cleft [50]. Hardware realization of the synapse-population model behaves as expected [Fig. 9(a)–(d)].

The conductance $g_{\rm syn}$ decays spatially in the shared dendritic tree and provides an input current

$$\zeta(n)g_{\rm syn}(e_{\rm syn} - v_s) \quad \text{or} \quad \zeta(n)g_{\rm syn}(e_{\rm syn} - v_d) \qquad (6)$$

to the soma or dendrite, respectively (in addition to $i_{\rm sin}$ or $i_{\rm din}$). Here

$$\zeta(n) = \frac{1}{4\sqrt{\pi}}\left(1 + \left(\frac{1}{1-\gamma^2}\right)^{\frac{1}{4}}\right)^2 \frac{\gamma^n}{\sqrt{n}} \qquad (7)$$

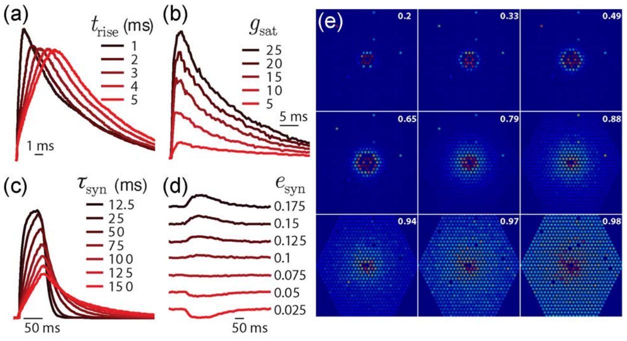


Fig. 9. ***Synapse population circuit's conductance traces. (a) Increasing t_{rise}: It prolongs the rising phase. To keep the area constant, g_{sat} was divided by t_{rise}. (b) Increasing g_{sat}: It increases the synaptic conductance proportionately. (c) Increasing τ_{syn}: It slows integration, resulting in smaller peak conductances and longer decay times. (d) Increasing e_{syn}: It changes the effect on the membrane potential from inhibitory (red) to excitatory (black). (e) Increasing γ (left to right, top to bottom): It increases the spread of synaptic conductances evoked at six locations, arranged in a hexagon, by a spike delivered to their center.***

where γ is the silicon dendritic tree's decay factor and n is the distance traveled in a number of neurons [51]. Hardware realization of the dendritic tree behaves as expected [Fig. 9(e)].

Ion-Channel Population: The ion-channel population's conductance g_{ch} is obtained by scaling a maximum conductance $g_{\max}$ with a gating variable c (i.e., $g_{\mathrm{ch}} = cg_{\max}$; see Fig. 6 Channel Population). c is modeled as

$$\tau_{\mathrm{gv}}\dot{c} = -c + c_{\mathrm{ss}} \tag{8}$$

where c_{ss} is its steady-state activation or inactivation and τ_{ch} is its time constant. c_{ss} is given by

$$c_{\mathrm{ss}} = \frac{\alpha}{\alpha+\beta} \quad \text{or} \quad \frac{\beta}{\alpha+\beta} \tag{9}$$

where α and β model a channel's opening and closing rates, whose voltage dependence is modeled as

$$\alpha, \beta = \pm\frac{1}{2}(v_{\mathrm{d}} - v_{\mathrm{th}}) + \frac{1}{2}\sqrt{(v_{\mathrm{d}} - v_{\mathrm{th}})^2 + \frac{1}{4s^2}}. \tag{10}$$

Here, v_{th} is the membrane potential at which $c_{\mathrm{ss}} = 1/2$ and s is the slope at this point. α and β satisfy a difference relation $\alpha - \beta = v_{\mathrm{d}} - v_{\mathrm{th}}$ and a reciprocal relation $\alpha\beta = 1/(16s^2)$, resulting in a sigmoidal dependence of c_{ss} on v_{d}. The gating variable's time constant is given by

$$\tau_{\mathrm{gv}} = \frac{\tau_{\max} - \tau_{\min}}{2s(\alpha+\beta)} + \tau_{\min}. \tag{11}$$

τ_{gv} is bell shaped with a maximum value of $\tau_{\max}$ when $v_{\mathrm{d}} = v_{\mathrm{th}}$ and a minimum value of $\tau_{\min}$ when $|v_{\mathrm{d}} - v_{\mathrm{th}}| \gg 1/(2s)$, to avoid unphysiologically short time constants. Hardware realization of the ion-channel population model behaves as expected (Fig. 10).

B. Circuit Realization of Dimensionless Models

Before describing the dimensionless models' circuit implementations, we illustrate how such models may be realized in the log domain [33] by using MOS devices operating in the subthreshold regime to realize a passive membrane [see (1)].

In the subthreshold regime, a PMOS transistor's drain current I_{d} is related to its gate-bulk voltage V_{gb} by

$$I_{\mathrm{d}} = \Lambda I_0 e^{-\frac{\kappa V_{\mathrm{gb}} - V_{\mathrm{sb}}}{U_{\mathrm{T}}}}\left(1 - e^{\frac{V_{\mathrm{ds}}}{U_{\mathrm{T}}}}\right) \tag{12}$$

where $\Lambda = W/L$ is the transistor's width-to-length ratio, I_0 is the leakage current when $\Lambda = 1$, κ is the ratio between effective and applied gate voltage, and U_{T} is the thermal voltage[8]; and V_{sb} and V_{ds} are the source-bulk and drain-source voltages, respectively [38]. For $V_{\mathrm{ds}} < -4U_{\mathrm{T}}$ and $V_{\mathrm{sb}} = 0$, (12) is approximately

$$I_{\mathrm{d}} = \Lambda I_0 e^{-\frac{\kappa V_{\mathrm{gb}}}{U_{\mathrm{T}}}}.$$

Taking natural logarithm on both sides

$$\ln I_{\mathrm{d}} - \ln I_0 - \ln\Lambda = -\frac{\kappa V_{\mathrm{gb}}}{U_{\mathrm{T}}}. \tag{13}$$

Differentiating (13) with respect to time, we obtain

$$\frac{\dot{I}_{\mathrm{d}}}{I_{\mathrm{d}}} = -\frac{\kappa}{U_{\mathrm{T}}}\dot{V}_{\mathrm{gb}}. \tag{14}$$

These equations are the basis for realizing models in the log domain with MOS transistors.

The passive membrane's circuit realization (Fig. 11) consists of a capacitor (C), with the voltage across it (V_{m}) driving a transistor (M_5) to produce an output current (I_{m}) that represents the membrane's potential. A current sourced into the capacitor (I_{lk}) represents the membrane's leak and a current sunk from the capacitor (I_{back}) represents the membrane's input. Kirchoff's current law gives

$$C\dot{V}_{\mathrm{m}} = I_{\mathrm{lk}} - I_{\mathrm{back}}. \tag{15}$$

[8] $U_{\mathrm{T}} = kT/q$, where k is the Boltzmann constant, T is the absolute temperature, and q is an electron's charge.

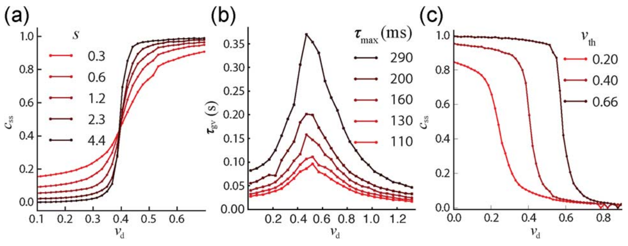


Fig. 10. *Gating-variable curves from ion-channel population circuit. (a) Steady-state value increases (for activation) with increasing membrane potential, exhibiting a sigmoidal dependence. Increasing s increases the slope. (b) Time constant has a bell-curved dependence on membrane potential.* $\tau_{\max}$ *scales the peak. (c) Threshold increases with increasing* v_{th}.

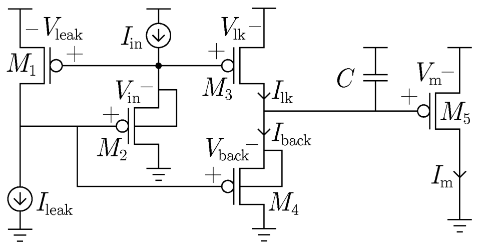


Fig. 11. *Passive membrane circuit. It models the membrane's time constant* τ *(through* I_{leak}*), input current* u *(through* I_{in}*), and potential* v *(through* I_{m}*). Bulk terminals are connected to the power rail unless otherwise indicated.*

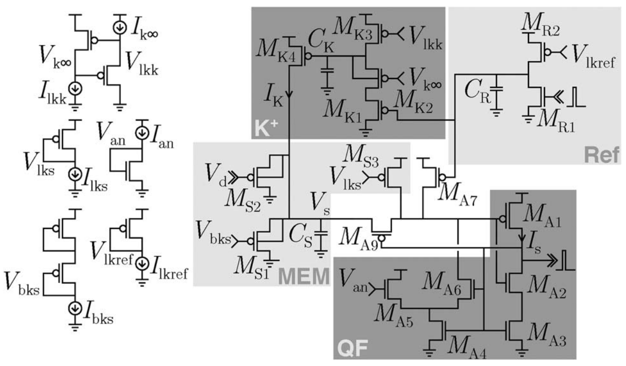


Fig. 12. ***Soma circuit. MEM models membrane time constant τ_s (through I_{lks}), input current i_{sin} (through I_{bks}), and dendritic input v_d (through I_d; see dendrite circuit). QF models quadratic feedback $v_s^2/2$ (through I_{an}). K$^+$ models high-threshold potassium conductance (through I_{lkk} and $I_{k\infty}$). Ref models reset conductance g_{res} and refractory pulse p_{res} (through I_{lkref}).***

As M_1's and M_3's gate-bulk voltages are equal, we have

$$V_{leak} = V_{lk} \Rightarrow \frac{I_{leak}}{\Lambda_1} = \frac{I_{lk}}{\Lambda_3}. \tag{16}$$

As the sum of M_1's and M_2's gate-bulk voltages is equal to the sum of M_4's and M_5's, we have

$$V_{leak} + V_{in} = V_{back} + V_m \Rightarrow \frac{I_{leak}}{\Lambda_1}\frac{I_{in}}{\Lambda_2} = \frac{I_{back}}{\Lambda_4}\frac{I_m}{\Lambda_5}. \tag{17}$$

Using (14), (16), and (17) in (15) yields

$$\begin{aligned}\frac{CU_T}{\kappa}\frac{\dot{I}_m}{I_m} &= -\frac{\Lambda_3}{\Lambda_1}I_{leak} + \frac{\Lambda_4\Lambda_5}{\Lambda_1\Lambda_2}\frac{I_{leak}I_{in}}{I_m}\\ \Rightarrow \underbrace{\frac{p_\tau}{I_{leak}}}_{\tau}\underbrace{\frac{\dot{I}_m}{I_{leak}}}_{\dot{v}} &= -\underbrace{\frac{I_m}{I_{leak}}}_{v} + \underbrace{p_{in}\frac{I_{in}}{I_{leak}}}_{u}\end{aligned} \tag{18}$$

where $p_\tau = CU_T\Lambda_1/\kappa\Lambda_3$ and $p_{in} = \Lambda_4\Lambda_5/\Lambda_2\Lambda_3$ are the mapping constants required to program I_{leak} and I_{in} to realize the desired values of τ and u.[9]

Soma: The soma model's circuit realization (Fig. 12) operates according to

$$\overbrace{\frac{p_{\tau_s}}{I_{lks}}}^{\tau_s}\overbrace{\frac{\dot{I}_s}{I_{ns}}}^{\dot{v}_s} = -\overbrace{\frac{I_s}{I_{ns}}}^{v_s} + \overbrace{p_{qua}\frac{I_{bks}^2}{I_{lks}^2}}^{i_{sin}} + \overbrace{\frac{1}{2}\frac{I_s^2}{I_{ns}^2}}^{\frac{v_s^2}{2}} - \overbrace{\lambda_K\frac{I_K}{I_{lks}}}^{g_K}\overbrace{\frac{I_s}{I_{ns}}}^{v_s}$$
$$-\underbrace{\lambda_{res}\frac{I_{VDD}}{I_{lks}}}_{g_{res}}\underbrace{\frac{I_s}{I_{ns}}}_{v_s}p_{res}(t) + \underbrace{\frac{I_d}{I_{nd}}}_{v_d}$$

[9]To make τ independent of temperature, we used a proportional-to-absolute-temperature (PTAT) current source to generate I_{leak} [52], [53], which keeps U_T/I_{leak} constant.

Table 4 Mapping Constants

Soma and Dendrite						
p_{τ_s}		k_s	p_{qua}		p_{ref}	λ_{res}
$\frac{C_sU_T}{\kappa\Lambda_{S3}}$		$\frac{2\Lambda_{A4}\Lambda_{A6}\Lambda_{A1}}{\Lambda_{S3}\Lambda_{A3}(\Lambda_{A4}+\Lambda_{A6})}$	$k_s\frac{\Lambda_{S1}}{\Lambda_{S3}}$		$\frac{C_R\text{VDD}}{\Lambda_{R2}}$	$\frac{\Lambda_{A7}}{\Lambda_{S3}}$
p_{τ_K}	λ_K	k_d	p_{gK}	p_{τ_d}	p_{bkd}	p_{bp}
$\frac{C_KU_T}{\kappa\Lambda_{K3}}$	$\frac{\Lambda_{K4}}{\Lambda_{S3}}$	$k_s\frac{\Lambda_{S2}}{\Lambda_{S3}}$	$\frac{\Lambda_{K2}}{\Lambda_{K3}}$	$\frac{C_DU_T}{\kappa\Lambda_{M2}}$	$\frac{\Lambda_{M1}}{\Lambda_{M2}}$	$\frac{\Lambda_{B2}}{\Lambda_{M2}}$
Synapse						
$p_{\tau_{syn}}$		λ_{gsyns}	λ_{gsynd}	p_{gsat}	p_{esyn}	p_c
$\frac{C_RU_T}{\kappa\Lambda_{R3}}$		$\frac{\Lambda_{D4}\Lambda_{G5}}{\Lambda_{D5}\Lambda_{G1}\Lambda_{S3}}$	$\lambda_{gsyns}\frac{\Lambda_{S3}}{\Lambda_{M2}}$	$\frac{\Lambda_{R2}\Lambda_{R4}}{\Lambda_{R3}}$	$\frac{\Lambda_{G3}\Lambda_{G4}}{\Lambda_{G2}\Lambda_{G5}}$	$\frac{C_C\text{VDD}}{\Lambda_{C2}}$
Gating Variable†						
p_{gvmax}	p_{th}	$p_{\tau_{gv}}$	p_{slope}	$p_{\tau_{gvmax}}$	λ_{gch}	p_{ech}
$\frac{\Lambda_{C1}\Lambda_{T5}}{\Lambda_{T1}}$	$\frac{\Lambda_{O1}}{\Lambda_{O5}}$	$\frac{C_GU_T\Lambda_{V1}\Lambda_{V2}\Lambda_{T5}}{\kappa\Lambda_{C5}\Lambda_{V3}\Lambda_{T1}}$	$\frac{\sqrt{\Lambda_{N1}\Lambda_{N2}}}{\Lambda_{O5}}$	$\frac{p_{\tau_{gv}}}{2\sqrt{\Lambda_{N1}\Lambda_{N2}}}$	$\frac{\Lambda_{AM}\Lambda_{E5}}{\Lambda_{I5}\Lambda_{M2}}$	$\frac{\Lambda_{I5}}{\Lambda_{E5}}$

†$\Lambda_{N1} = \Lambda_{O2,3,6,7}$, $\Lambda_{N2} = \Lambda_{O4,8}$, $\Lambda_{AM} = \Lambda_{I1\text{-}4,E1\text{-}4}$.

which is equivalent to (2). Here

$$I_{ns} = \frac{I_{lks}}{k_s}, \quad I_{nd} = \frac{I_{lks}^2}{k_dI_0}, \quad \text{and} \quad t_{res} = \frac{p_{ref}}{I_{lkref}}$$

and I_{VDD} is the drain current of a PMOS with gate grounded and source at the supply rail. All mapping constants are defined in Table 4 (in this table, Λ_n is the W/L of transistor M_n). The high-threshold potassium conductance's circuit realization operates according to

$$\overbrace{\frac{p_{\tau_K}}{I_{lkk}}}^{\tau_K}\overbrace{\frac{\dot{I}_K}{I_{nK}}}^{\dot{g}_K} = -\overbrace{\frac{I_K}{I_{nK}}}^{g_K} + p_{gK}\overbrace{\frac{I_{K\infty}}{I_{nK}}}^{g_{K\infty}}p_{res}(t)$$

which is equivalent to (3). Here, $I_{nK} = I_{lks}/\lambda_K$. Detailed soma-circuit descriptions may be found elsewhere [54], [55].

Dendrite: The dendrite model's circuit realization (Fig. 13) operates according to

$$\overbrace{\frac{p_{\tau_d}}{I_{lkd}}}^{\tau_d}\overbrace{\frac{\dot{I}_d}{I_{nd}}}^{\dot{v}_d} = -\overbrace{\frac{I_d}{I_{nd}}}^{v_d} + \overbrace{p_{bkd}\frac{I_{bkd}^2}{I_{nd}I_{lkd}}}^{i_{din}} + \overbrace{p_{bp}\frac{I_{bp}^2}{I_{nd}I_{lkd}}}^{i_{bp}}p_{res}(t)$$

which is equivalent to (4).

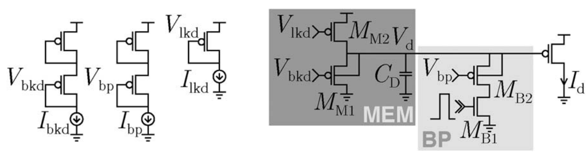


Fig. 13. ***Dendrite circuit. MEM models membrane time constant τ_d (through I_{lkd}) and input current i_{din} (through I_{bkd}). BP models backpropagating input i_{bp} (through I_{bp}).***

Synapse Population: The synapse population model's circuit realization (Fig. 14) operates according to

$$\overbrace{\frac{p_{\tau_{\mathrm{syn}}}}{I_{\mathrm{lklpf}}}}^{\tau_{\mathrm{syn}}}\overbrace{\frac{\dot{I}_{\mathrm{gsyn}}}{I_{\mathrm{ngsyns}}}}^{\dot{g}_{\mathrm{gsyn}}} = -\overbrace{\frac{I_{\mathrm{gsyn}}}{I_{\mathrm{ngsyns}}}}^{g_{\mathrm{syn}}} + \overbrace{p_{\mathrm{gsat}}\frac{I_{\mathrm{gsat}}}{I_{\mathrm{ngsyns}}}}^{g_{\mathrm{sat}}} p_{\mathrm{rise}}(t)$$

where $t_{\mathrm{rise}} = p_c/I_{\mathrm{lkpe}}$ and $I_{\mathrm{ngsyns}} = I_{\mathrm{lks}}/\lambda_{\mathrm{gsyns}}$.

When fed to the soma or dendrite circuit through the dendritic-tree and reversal-potential subcircuits, I_{gsyn} yields the (normalized) synaptic input to soma or dendrite as

$$\overbrace{\frac{I_{\mathrm{gsyn}}}{I_{\mathrm{ngsyns,d}}}}^{g_{\mathrm{syn}_i}}\left(\overbrace{p_{\mathrm{esyn}}\frac{I_{\mathrm{esyn}}}{I_{\mathrm{ns,d}}}}^{e_{\mathrm{syn}_i}} - \overbrace{\frac{I_{\mathrm{s,d}}}{I_{\mathrm{ns,d}}}}^{v_{\mathrm{s,d}}}\right)$$

which is equivalent to (6). Here, $I_{\mathrm{ngsynd}} = I_{\mathrm{lkd}}/\lambda_{\mathrm{gsynd}}$. A detailed synapse-circuit description may be found elsewhere [32].

The dendritic-tree circuit spreads the current I_{gsyn} through a hexagonal resistive network implemented with transistors [36]. Its decay factor γ is related to the voltages V_{r} and V_{g} as

$$\gamma = 1 - \frac{2}{1+\sqrt{1+8e^{\frac{\kappa(V_{\mathrm{r}}-V_{\mathrm{g}})}{U_{\mathrm{T}}}}}}.$$

Ion-Channel Population: The ion-channel population model's circuit realization (Fig. 15) computes the channel's conductance (g_{ch}) directly instead of computing the gating variable first (8) and then scaling the saturation conductance (g_{max}) by this. That is, the circuit operates as follows:

$$\overbrace{\frac{p_{\tau_{\mathrm{gv}}}I_{\mathrm{max}}}{(I_\alpha+I_\beta)I_{\mathrm{shift}}}}^{\tau_{\mathrm{gv}}}\overbrace{\frac{\dot{I}_{\mathrm{gv}}}{I_{\mathrm{ngv}}}}^{\dot{g}_{\mathrm{ch}}} = -\overbrace{\frac{I_{\mathrm{gv}}}{I_{\mathrm{ngv}}}}^{g_{\mathrm{ch}}} + \overbrace{\frac{I_\alpha}{I_\alpha+I_\beta}}^{c_{\mathrm{ss}}}\overbrace{p_{\mathrm{gvmax}}\frac{I_{\mathrm{max}}}{I_{\mathrm{ngv}}}}^{g_{\mathrm{max}}}$$

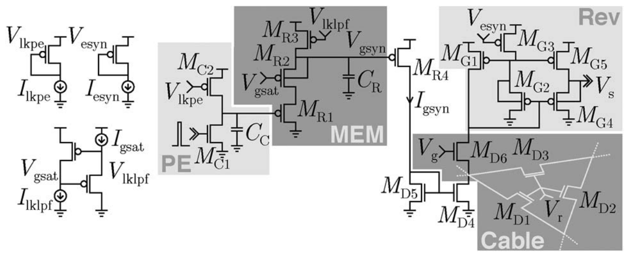


Fig. 14. ***Synapse-population circuit. PE models the rise time t_{rise} (through I_{lkpe}). MEM models time constant τ_{syn} (through I_{lklpf}) and saturation conductance g_{sat} (through I_{gsat}). Cable models spatial decay factor γ (through V_{r}). Rev models reversal potential e_{syn} (through I_{esyn}).***

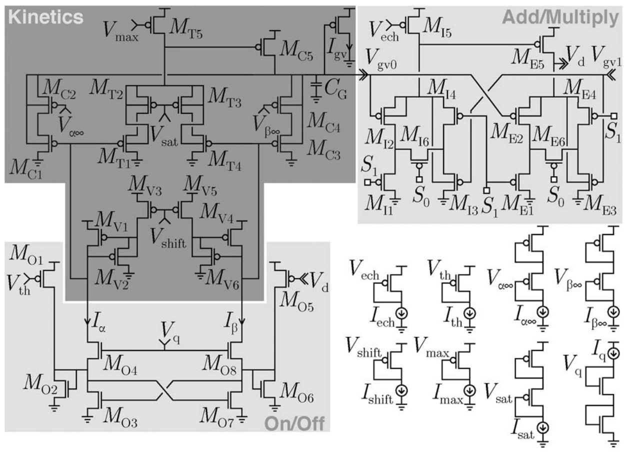


Fig. 15. ***Ion-channel population circuit.*** ***ON–OFF models opening and closing rates α and β (through I_α and I_β, respectively). Kinetics models peak time constant τ_{max} (through I_{shift}), activating/inactivating behavior of c_{ss} (through $I_{\alpha\infty}$ and $I_{\beta\infty}$), and maximum conductance g_{max} (through I_{max}). Add/multiply combines the output of two kinetics subcircuits to model their equivalent series or parallel conductance (when $[S_0, S_1] = [0, 1]$ or $[1, 0]$, respectively) and models reversal potential e_{ch} (through I_{ech}).***

for activating behavior (i.e., $I_{\alpha\infty} \to \infty$ and $I_{\beta\infty} \to 0$) and $\tau_{\mathrm{min}} = 0$ (i.e., $I_{\mathrm{sat}} \to \infty$). Here, $I_{\mathrm{ngv}} = I_{\mathrm{lkd}}I_{\mathrm{ech}}/(\lambda_{\mathrm{gch}}I_0)$. I_{shift} realizes τ_{max} as $p_{\tau_{\mathrm{gv\,max}}}I_{\mathrm{max}}/I_{\mathrm{q}}I_{\mathrm{shift}}$. I_α and I_β realize α and β as

$$\overbrace{\frac{I_{\alpha,\beta}}{\Lambda_{\mathrm{O5}}I_{\mathrm{nd}}}}^{\alpha,\beta} \approx \pm\frac{1}{2}\left(\overbrace{\frac{I_{\mathrm{d}}}{I_{\mathrm{nd}}}}^{v_{\mathrm{d}}} - \overbrace{p_{\mathrm{th}}\frac{I_{\mathrm{th}}}{I_{\mathrm{nd}}}}^{v_{\mathrm{th}}}\right) + \frac{1}{2}\sqrt{\left(\underbrace{\frac{I_{\mathrm{d}}}{I_{\mathrm{nd}}}}_{v_{\mathrm{d}}} - \underbrace{p_{\mathrm{th}}\frac{I_{\mathrm{th}}}{I_{\mathrm{nd}}}}_{v_{\mathrm{th}}}\right)^2 + \underbrace{4\left(p_{\mathrm{slope}}\frac{I_{\mathrm{q}}}{I_{\mathrm{nd}}}\right)^2}_{1/4s^2}}$$

for $I_\alpha \gg I_\beta$ or *vice versa*, which is equivalent to (10). Unlike a previous implementation [56], this one supports adjustable slopes for activation and inactivation through an ON–OFF circuit [57].

Another circuit combines I_{gv} from a pair of circuits ($I_{\mathrm{gv}_{0,2}}$ and $I_{\mathrm{gv}_{1,3}}$) to obtain their conductances' series or parallel combinations (g_{ch_0} and g_{ch_1}; see Fig. 6), modeled as

$$I_{\mathrm{ch}_{0,1}} = \begin{cases} \dfrac{I_{\mathrm{gv}_{0,2}}I_{\mathrm{gv}_{1,3}}}{I_{\mathrm{gv}_{0,2}}+I_{\mathrm{gv}_{1,3}}}, & \text{series} \\ I_{\mathrm{gv}_{0,2}}+I_{\mathrm{gv}_{1,3}}, & \text{parallel} \end{cases}$$

which drives the dendrite with a current

$$\overbrace{\frac{I_{\mathrm{ch}_{0,1}}}{I_{\mathrm{ngv}}}}^{g_{\mathrm{ch}_{0,1}}}\left(\overbrace{p_{\mathrm{ech}}\frac{I_{\mathrm{ech}}}{I_{\mathrm{nd}}}}^{e_{\mathrm{ch}_{0,1}}} - \overbrace{\frac{I_{\mathrm{d}}}{I_{\mathrm{nd}}}}^{v_{\mathrm{d}}}\right).$$

The series combination is used to model channels that activate and inactivate (second order), while the parallel combination models independent channels (first order) with a common reversal potential.

To correct errors in the mapping constants' analytical expressions (see Table 4) due to deviations from the transistor model (12) [58], we calibrated them by measuring four types of circuit responses (Table 5): 1) dynamic current (e.g., the dendrite's exponentially decaying response to a step input, which we used to calibrate p_{τ_d}); 2) steady-state current (e.g., the ion-channel population's conductance for a given dendritic potential, which we used to calibrate p_{th}); 3) steady-state spike rate (e.g., the linear scaling of a neuron's spike rate with its time constant, which we used to calibrate p_{τ_s} [54]); and 4) spike-rate discontinuity (e.g., the onset of spiking when the dimensionless input exceeds 0.5, which we used to calibrate p_{qua} [54]). These procedures yielded mapping constants for individual circuits. We only used the median of these distributions, which arise from transistor mismatch, as all of a Neuorocore's neurons share the same biases.

V. TRANSMITTER, RECEIVER, AND ROUTER

A neuron's spike is dispatched from its array by a transmitter, communicated to its Neurocore's parent and two children by a router, and delivered to the recipient by a receiver. All this digital circuitry is event driven—only active when a spike occurs—with its logic synthesized following Martin's procedure for asynchronous circuits [60], [61].

A. Transmitter and Receiver

We provide brief descriptions of the transmitter's and receiver's architecture and operation; detailed descriptions may be found in [59] and [62]–[64]. The transmitter dispatches multiple spikes from a row and the receiver delivers multiple spikes to a row, enhancing throughput compared to designs that dispatch or deliver spikes one by one [19]–[21]. These spikes' common row address and unique column addresses are communicated sequentially.

In the transmitter [Fig. 16(a)], two M-way arbiters, built with $M-1$ two-way arbiters connected in a binary tree, receive requests from an array of spiking neurons with M rows and M columns. Only $\log_2(M)$ logic levels are

Table 5 Responses Used to Calibrate Mapping Constants

Dynamic Current	p_{τ_d}, $p_{\tau_{\text{syn}}}$, p_c, $p_{\tau_{\text{gvmax}}}$
Steady-State Current	p_{th}, p_{slope}
Steady-State Spike-Rate	p_{τ_s}, p_{ref}, p_{τ_K}, p_{gK}
Spike-Rate Discontinuity	p_{qua}, p_{bkd}, p_{gsat}, p_{esyn}, p_{gvmax}, p_{ech}

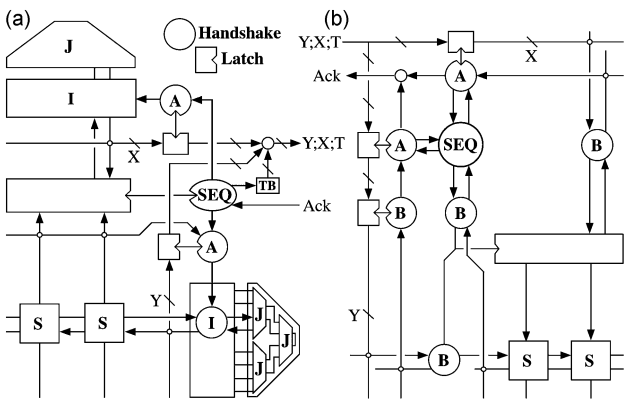


Fig. 16. *Transmitter and receiver architecture. (a) Transmitter: An interface (I) relays requests from spiking neurons (S) to a row arbiter (J) and dispatches the selected row's spikes (S) in parallel while encoding its address (Y). Another interface (I) relays the spikes from a latch to a column arbiter (J) and encodes the selected column's address (X). A sequencer (SEQ) directs latches (A) to deliver the row address, column address(es), and a tailword (T, generated by TB) to the output port. (b) Receiver: A sequencer (SEQ) directs two different latches (A) to load incoming row (Y) and column (X) addresses, which are decoded to select a row and one or more columns. These select lines are activated simultaneously, when the tailword (T) is received, delivering spikes to the row in parallel (S). The remaining latches operate autonomously (B), automatically overwriting old data after it has been read. Small discs symbolize combinational logic. Modified from [59].*

traversed to select a row or column, compared to $M/2$, on average, for a scanner [65], [66]. Two encoders generate a $\log_2(M)$-bit address for each row or column selected. Latches enable pipelining: The next row's spikes are dispatched from the array while the current row's column addresses are being encoded and sent out. In the receiver [Fig. 16(b)], these $\log_2(M)$-bit addresses are decoded to select one of M rows or columns. Again, latches allow the next packet's addresses to be decoded while the current one's spikes are being delivered to the array.

Neurocore uses a 256×256 version of the transmitter and a 2048×256 version of the receiver—its eight lines per row select one of four shared-synapses to activate, one of three sets of analog signals to sample, or a neuron to disable (all in conjunction with a column line). The transmitter takes 86 ns to transfer a row's spikes to the array's periphery. Then, it takes 23 ns to encode each spike's column address. Therefore, if there are three or more spikes, no time is wasted waiting for the next row's spikes to be transferred. Thus, pipelining and parallelism enable the transmitter to sustain a maximum transmission rate of 43.4 Mspike/s, or 663 spike/s per neuron [42]. The receiver decodes an additional column address every 16 ns, sustaining a maximum rate of 62.5 Mspike/s, or 956 spike/s per neuron [59].[10]

[10]This paper reports measurements from a 960×320 version of the receiver fabricated in the same technology (180-nm CMOS).

B. Router

We provide brief descriptions of the router's logical and physical design; detailed descriptions may be found in [42]. The router's multicast capability and the Neurocores' embedded memory realize secondary axon-branching (limited to corresponding locations in multiple Neurocores); primary axon-branching (to arbitrary locations in multiple Neurocores) is realized by the daughterboard's FPGA and SRAM.

The router routes a packet from a source to multiple recipients in two distinct phases: a point-to-point phase and a branching phase (Fig. 17). During the point-to-point phase, the router steers the packet up/down or left/right, based on a bit in the packet's first word. During the branching phase, the router copies the packet to both left and right ports. These flooding packets are delivered to the local neural array or filtered using information retrieved from a location in the Neurocore's 256 × 16-b SRAM, specified by the packet's second word. This approach achieves high throughput by distributing the packet to all its potential recipients without performing any memory lookups [27]. These lookups, which are the slowest operation, proceed in parallel, in contrast to a network where lookups decide the packet's route [6]. If the packet is delivered, two bits retrieved from the SRAM are appended to its row address. These bits specify which one of the recipient's four shared synapses to activate [27], [42].

The router's datapath consists of four merges and three splits (Fig. 18). For energy efficiency, the 12-b datapath was sliced into six bit-pairs, each communicated by two transitions—the second returns to zero—on one of four lines. This 1-in-4 code requires half as many transitions as independent bits require (1-in-2 code). Furthermore, a single acknowledge (or enable) line is used, instead of two [61]. The datapath's blocks were compact enough to be distributed throughout the IO-pad ring (see Fig. 5). Each port's pads were mostly placed on its corresponding side of the chip, which facilitated building a multichip printed circuit board with straight connections between adjacent chips. Two-to-one multiplexing cut the number of pads per port from 42 to 21, organized in three groups of seven pads (two power, four signal, and one enable). Each pad group transmits two bits with a single transition, without returning to zero, on one of four lines (1-change-in-4 code) [67], [68], achieving a data-rate of 364 Mb/s. Thus, each port can handle 91 Mword/s [42].

When interconnected in a binary tree, the router delivered 1.17 Gword/s to Neurogrid's 16 Neurocores with no more than 1-μs jitter along the longest path [42]. Jitter is defined as the standard deviation of intervals between packets injected at equal intervals. These injected packets (generated by a computer) were routed all the way up and down the tree, ejected at a leaf, and captured by a logic analyzer. Meanwhile, the eight Neurocores at the tree's leaves each generated spike packets at a rate of 9.14 Mword/s. This traffic aggregated at the root, which received 73.12 Mword/s in total, and flowed to all 16 Neurocores, which received 1.17 Gword/s in total. This delivery rate corresponds to 234 Mspike/s in normal mode (five words per spike packet) and up to 1.17 Gspike/s in burst mode (additional column addresses are appended to the packet).

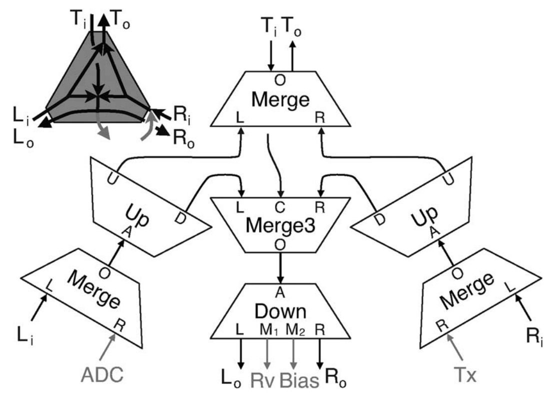


Fig. 18. ***Router datapath. Merge and Merge3 feed packets to splits: Up steers packet to its U or D port and Down steers packets to its L or R port. When it encounters a stop code, Up deletes the packet and Down interrogates the mode bit. If it is clear (targeted mode), it delivers the packet to either its M_1 or M_2 port (a bit in the headword decides). If it is set (flood mode), it delivers the packet to its L_o and R_o ports as well. Reproduced from [42].***

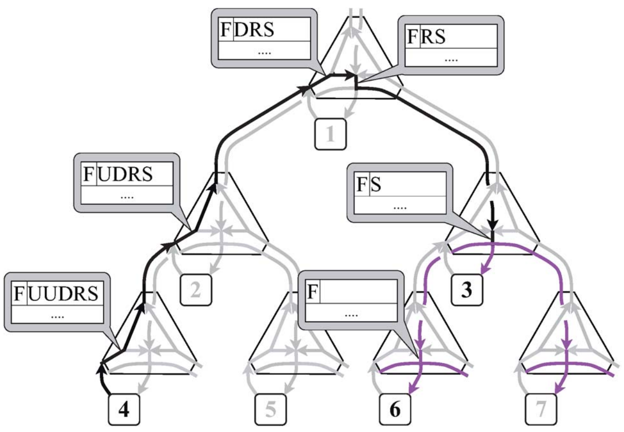


Fig. 17. ***Multicast routing's point-to-point and branching phases. In the point-to-point phase (black), node 4's packet is routed up to node 1, the lowest common ancestor node 4 shares with the recipients (nodes 3 and 6). The packet is then routed down to node 3, the recipients' lowest common ancestor. At each node, the route field's most significant bit [encodes the turn (U or D, R or L)] is shifted out and 0 is shifted in. The stop code [encodes the terminus (node 3)] is all zeroes except the MSB (S). In the branching phase (purple), the packet visits node 3 and all its descendants—node 7's SRAM is programmed to filter the packet. A mode bit (F) determines whether the packet floods or targets the terminus. Reproduced from [42].***

VI. ENERGY EFFICIENCY

We measured and dissected Neurogrid's energy consumption for a million-neuron, eight-billion-synapse real-time simulation and found that it uses energy more efficiently than expected from its shared-dendrite architecture.

Neurogrid was programmed to simulate a recurrent inhibitory network with 15 layers (Fig. 19), each mapped onto a different Neurocore. The recurrent synaptic connections were realized by multicasting spikes from each Neurocore to all others (including itself). That Neurocore and its three neighbors on either side where programmed to accept its spikes, which inhibited nearby neurons through the shared dendrite [with $\gamma = 0.94$; see Fig. 9(e)]. As a result, each of the model's 983 040 neurons received 50% of its inhibition from 7980 neurons residing in a seven-layer-thick, 19-neuron-radius cylinder centered around it. Such recurrent inhibitory connectivity patterns are expected to give rise to globally synchronous spike activity, which is what we observed. The synchronized activity was rhythmic, with a frequency of 3.7 Hz; the neurons fired 0.42 spike/s on average.

Neurogrid consumed 2.7 W during the simulation. Since interlayer connections were between corresponding locations (i.e., columnar), the daughterboard was not needed. Measurements of its power consumption from a separate study, where we demonstrate the ability to implement arbitrary connectivity patterns [69], revealed that it would consume 0.4 W to route the 413 000 spike/s that the 983 040 neurons produced, yielding a total of 3.1 W, or 941 pJ per synaptic activation (for 7980 synaptic connections per neuron).

The energy expended to activate a silicon axon's synapses (E_{axon}) may be expressed as the sum of the energy used (per spike) by the soma (E_{soma}), transmitter (E_{xmt}), up route (E_{u}), daughterboard (E_{RAM}), down route (E_{d}), and receiver (E_{rcv}); the last term includes the shared-dendrite and synapse circuits. That is

$$E_{\text{axon}} = E_{\text{soma}} + E_{\text{xmt}} + E_{\text{u}} + n_{\text{col}}(E_{\text{RAM}} + E_{\text{d}} + n_{\text{p}}E_{\text{rcv}}).$$

Here the axon connects to n_{p} pools of neurons (secondary branches) in each of n_{col} locations (primary branches). Therefore, the total number of synapses (n_{axon}) it activates is $n_{\text{col}} \times n_{\text{p}} \times n_{\text{syn}}$, where n_{syn} is the average number of synaptic connections per pool. Hence, the energy per synaptic activation ($E = E_{\text{axon}}/n_{\text{axon}}$) is

$$E = \frac{E_{\text{soma}} + E_{\text{xmt}} + E_{\text{u}}}{n_{\text{col}}n_{\text{p}}n_{\text{syn}}} + \frac{E_{\text{RAM}} + E_{\text{d}}}{n_{\text{p}}n_{\text{syn}}} + \frac{E_{\text{rcv}}}{n_{\text{syn}}}. \tag{19}$$

Terms associated with the axon's trunk ($E_{\text{soma}}, E_{\text{xmt}}, E_{\text{u}}$) and primary ($E_{\text{RAM}}, E_{\text{d}}$) and secondary ($E_{\text{rcv}}$) branches contribute small, medium, and large amounts, respectively. These energy terms (Table 6) were determined as follows.

Table 6 Energy Per Spike

Operation	E_{soma}	E_{xmt}	E_{u}	E_{RAM}	E_{d}	E_{rcv}
Energy (nJ)	$2612T_{\text{soma}}$	11.0	11.2	2.2	7.6	9.8

E_{soma} equals $V_{\text{jack}}I_{\text{static}}T_{\text{soma}}/N_{\text{total}}$, where $V_{\text{jack}} = 3$ V is the power jack's voltage, I_{static} is its current when the neurons are quiescent,[11] T_{soma} is the average interspike interval (for which E is calculated), and $N_{\text{total}} = 2^{20}$ is the neuron count. I_{static} only approximately captured the analog circuitry's current draw, which has a weak dependence on T_{soma}.

$E_{\text{xmt,rcv}}$ equals $V_{\text{jack}}\Delta I_{\text{xmt,rcv}}/f_{\text{total}}$, where ΔI_{xmt} is the additional current drawn when Neurocores generate f_{total} spike/s (each one routes its spikes back to itself but its synaptic connectivity RAM filters them) and ΔI_{rcv} is the additional current drawn when one Neurocore, to which all the other's spikes are routed, is reprogrammed to stop filtering them.

$E_{\text{u,d}}$ equals $n_{\text{u,d}}E_{\text{link}}$, where $n_{\text{u,d}}$ is the number of links between a Neurocore and the daughterboard and E_{link} is the energy a link uses to transmit a five-word spike packet [67].[12] We set $n_{\text{u}} = 3.4$, the average number of links from a Neurocore to the daughterboard, and $n_{\text{d}} = 2.3$, one-seventh the number of links a packet traverses when it floods from the root (i.e., $n_{\text{p}} = 7$).

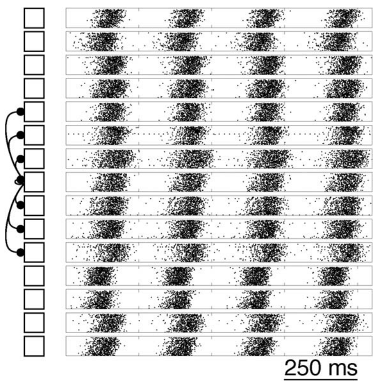


Fig. 19. ***Simulating a million neurons. The neurons were organized into fifteen 256 × 256 cell layers, arranged in a ring, so the first and last layers are nearest neighbors. Each cell layer's neurons inhibit neighboring neurons in its layer as well as in three neighboring layers to either side (the central layer's connectivity is shown). Spike rasters (from a tenth of each layer's neurons) reveal global synchrony, as expected from the network's recurrent inhibition.***

[11]The daughterboard drew 87 mA, the FX2 and CPLD drew 116 mA, the regulators (three per Neurocore) drew 74 mA, and the 16 Neurocores' IO, digital and analog circuitry drew 25, 64, and 572 mA, respectively. These measurements were made with an HP E3631A DC power supply.

[12]We scaled this measurement, which was made on a 3-cm-long link, to match the average length of the interneurcore links (1.1 cm), assuming that energy is proportional to length.

Table 7 Cost Comparison

	A (μm^2)	E (pJ)†	T (ps)	$A{\cdot}E{\cdot}T$	S	Synapse Spec
HICANN	436	198	2.9	250K	224	4-bit plastic
GoldenGate	256	4.0K	29.5	30.6M	1024	1-bit dedicated
Neurogrid	0.63	119	62.5	4.69K	4096	13-bit shared

$^\dagger E$ includes dynamic and static power (for a 10 spike/s/neuron simulation).

$E_{\rm RAM}$ equals $V_{\rm jack}\Delta I_{\rm lkup}/f_{\rm lkup}$, where $\Delta I_{\rm jack}$ is the additional current the daughterboard's power jack draws when it performs $f_{\rm lkup}$ memory lookups per second.

Substituting these measured values into (19) yields (in nJ)

$$E = \frac{22.2 + 2612 T_{\rm soma}}{n_{\rm col} n_{\rm p} n_{\rm syn}} + \frac{9.8}{n_{\rm p} n_{\rm syn}} + \frac{9.8}{n_{\rm syn}}. \qquad (20)$$

This model predicts $E =$ 813 pJ for the synchrony simulation, where $n_{\rm col} = 1$, $n_{\rm p} = 7$, $n_{\rm syn} = 1140$, and $T_{\rm soma} = 1/(0.42$ spike/s). This prediction is within 14% of the measured value of $E =$ 941 pJ. This discrepancy is probably due to ignoring the analog circuitry's slight increase in power dissipation with spike rate.

VII. *AET* COST COMPARISON

Neurogrid's overall cost (*AET*) is lower than HICANN's [9] and GoldenGate's [8] (Table 7).[13] We included the router's costs and the entire system's power dissipation in our A and E calculations, unlike in Section II-B. Nonetheless, our architecture-scaling model's predictions held up: A was smallest for Neurogrid (SDH), E was highest for GoldenGate (SAD), and T was smallest for HICANN (FDA). However, Neurogrid's A and E were smaller than predicted because it amortized these costs over a larger number of synaptic connections by using multilevel axon branching. And HICANN's T was larger than predicted because it did not realize FDA fully—it multiplexes 64 neurons' spikes onto each shared axon. As a result, Neurogrid's *AET* was 50 times lower than HICANN's. Before discussing these results in detail, we describe how A, E, and T were calculated.

A was calculated as $A_{\rm chip}/(SN_{\rm chip})$, where $A_{\rm chip}$ is the chip's area, $N_{\rm chip}$ is its neuron count, and S is the number of synapses (HICANN and GoldenGate) or the number of synaptic connections (Neurogrid) per neuron (see Table 7). In Neurogrid' case, we used $n_{\rm col} = 4$, $n_{\rm p} = 4$, and $n_{\rm syn} = 256$ (midrange values).

E was calculated as $P_{\rm sys}/(f_{\rm avg} N_{\rm sys} S)$, where $P_{\rm sys}$ is the system's total power at an average firing rate of $f_{\rm avg}$ and $N_{\rm sys}$ is its total neuron count. To account for static dissipation, we set $f_{\rm avg} = 10$ spike/s for all three systems; actually 10^5 spike/s for HICANN ($10^4\times$ speedup). $P_{\rm sys}$ was 800 W for HICANN's 352-chip, 48-FPGA, 180 000-neuron wafer-scale system [71] and 5 mW for GoldenGate's 256-neuron chip [8]. For Neurogrid, we obtained E using (20).[14]

T was calculated as $t_{\rm axon}/(n_{\rm par} n_{\rm axon})$, identical to Section II-B. $t_{\rm axon}$ was 5.21 ns for HICANN (i.e., 192 Mspike/s), 7.56 ns for GoldenGate (scaled from a 1-ns cycle-time estimate for a 45-nm SRAM), and 16.0 ns for Neurogrid. We used the receiver's word rate, since it is slower than the router's (13.7 ns at 1-μS jitter). $n_{\rm par}$ was 224 for HICANN and 1 for GoldenGate and Neurogrid. $n_{\rm axon}$ was 8 for HICANN and 256 for GoldenGate and Neurogrid (using shared dendrite).

To understand why Neurogrid's *AET* was lower than expected, consider the case where neither primary nor secondary axon branching is used (i.e., $n_{\rm p} = n_{\rm col} = 1$), which makes Neurogrid's S similar to HICANN's. The resulting 16-fold drop increases A 16-fold, increases E tenfold [using (20)] and leaves T unchanged. As a result, Neurogrid's *AET* becomes three times larger than HICANN's. Therefore, Neurogrid achieved lower than expected *AET* by utilizing multilevel axon branching to amortize its fixed area and static energy costs over more synaptic connections. With S matched, Neurogrid's E is six times larger than HICANN's, probably because Neurogrid's neuron has four shared-synapse and four shared-dendrite circuits—not one each—as well as four ion-channel circuits. Indeed, it is six times larger in area (2560 μm^2) than HICANN's synapse (see Table 7).

VIII. DISCUSSION

We found that Neurogrid's shared-dendrite (SD) architecture achieved the lowest *AET* cost by using multilevel axon branching to increase synaptic connectivity. Its bandwidth-efficient interarray communication mechanism—multicast tree router (TR)—made this possible. We conclude by discussing how large-scale neural models can fully exploit Neurogrid's simulation capacity, the limitations its architecture has, and how its configurability and scale may be increased.

Neurogrid's cost-effective SD architecture and TR topology can be fully exploited by neural models that satisfy two requirements. First, they are organized into layers such that neighboring neurons within the same layer have mostly the same inputs (as in cortical feature maps). Second, they are organized into columns such that neurons at corresponding locations in different layers have translation-invariant connectivity (as in cortical columns). The first requirement allows SD to be used. Otherwise, the receiver has to cycle $n_{\rm syn}$ times to deliver the spike to $n_{\rm syn}$ targets, instead of just once. The second requirement

[13]For comparison with HICANN and Neurogrid, we scaled GoldenGate's $A = 16$ μm^2, $E = 1.9$ nJ, and $T = 3.9$ pS from a 0.85 V–45 nm process to a 1.8 V–180 nm process using general scaling laws [70].

[14]We could not measure the power directly because USB throughput was limited to about a million spike/s (constrained by our current CPLD firmware).

allows multicast routing to be used. Otherwise, the daughterboard has to cycle $n_{\mathrm{col}} \times n_p$ times to route the spike to arbitrary locations, instead of just n_{col} times.

A limitation of Neurogrid's SD architecture is it precludes synaptic plasticity, which HICANN and other realizations of the fully dedicated architecture (FD) support [72], [73]. This limitation arises because neighboring neurons share the same (spatially decayed) input. Nonetheless, Neurogrid also supports the shared-synapse (SS) architecture, which does allow individual connection weights (stored in the daughterboard's RAM [69], [74]). SS can realize spike-timing-dependent plasticity (STDP), which HICANN realizes, by tracking a synapse's recent spike history (i.e., queuing address–events) and updating the stored weight accordingly [75]. However, SS is N times less *AET*-efficient than FD (see Table 1). In practice, however, FD's N^2 area scaling makes it prohibitively expensive to furnish each neuron with thousands of synapses—HICANN has only 224 synapses per neuron—until emerging nanoscale devices become viable [76].

Neurogrid's configurability and scale may be increased by migrating from its decade-and-a-half-old process to a state-of-the-art one, which will allow the memory embedded in each Neurocore to increase by two orders of magnitude. This additional memory will make it possible to replace shared dendrites with local shared axons (i.e., tertiary branches), which offer greater configurability while being area-efficient. And it will make it possible to route spikes to arbitrary locations without sending the packet all the way to the root, relieving traffic there and enabling the design to scale further.

In addition to increasing neurogrid's configurability and scale, the higher level of integration a state-of-the-art process offers will result in greater energy efficiency. While Neurogrid's energy efficiency is five orders of magnitude better than a personal computer's, it is four to five orders of magnitude worse than the human brain's. Neurogrid uses a few watts to simulate a million neurons in real time whereas a personal computer uses a few hundred watts to simulate 2.5 million neurons 9000 times slower than real time. The human brain, with 80 000 times more neurons than Neurogrid, consumes only three times as much power. Achieving this level of energy efficiency while offering greater configurability and scale is the ultimate challenge neuromorphic engineers face. ■

Acknowledgment

The authors would like to thank S. Menon for insightful suggestions on router design; C. Sauer and K. M Hynna for help with chip design; J. M. Knight for help with chip simulations; D. L. Neil for help with hardware testing and debugging; S. A. Elassaad for help with printed circuit board revision and software architecture; R. Roelofs for help with software optimization; and K. Chin, M. Phan, and J. L. Hollings for administrative support.

REFERENCES

[1] C. Eliasmith and O. Trujillo, "The use and abuse of large-scale brain models," *Current Opinion Neurobiol.*, vol. 25, pp. 1–6, 2014.

[2] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, C. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, Nov. 2012.

[3] S. Herculano-Houzel, "Scaling of brain metabolism with a fixed energy budget per neuron: Implications for neuronal activity, plasticity and evolution," *PLoS One*, vol. 6, no. 3, 2011, e17514.

[4] H. Markram, "The human brain project," *Sci. Amer.*, vol. 306, no. 6, pp. 50–55, May 2012.

[5] P. Kogge, "The tops in flops," *IEEE Spectrum*, vol. 48, no. 2, pp. 48–54, Feb. 2011.

[6] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.

[7] S. B. Furber, F. Gallupi, S. Temple, and L. A. Plana, "The SpiNNaker project: A massively-parallel computer architecture for neural simulations," *Proc. IEEE*, vol. 102, no. 5, May 2014, DOI: 10.1109/JPROC.2014. 2304638.

[8] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45 pJ per spike in 45 nm," in *Proc. Custom Integr. Circuits Conf.*, 2011, DOI: 10.1109/CICC.2011.6055294.

[9] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 1947–1950.

[10] R. Silver, K. Boahen, S. Grillner, N. Kopell, and K. L. Olsen, "Neurotech for neuroscience: Unifying concepts, organizing principles, emerging tools," *J. Neurosci.*, vol. 27, no. 44, pp. 11807–11819, 2007.

[11] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[12] C. Mead, *Analog VLSI and Neural Systems*. Boston, MA, USA: Addison-Wesley, 1989.

[13] C. Diorio, P. Hasler, B. Minch, and C. Mead, "A single-transistor silicon synapse," *IEEE Trans. Electron Devices*, vol. 43, no. 11, pp. 1972–1980, Nov. 1996.

[14] B. Linares-Barranco, T. Serrano-Gotarredona, and R. Serrano-Gotarredona, "Compact low-power calibration mini-DACs for neural arrays with programmable weights," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1207–1216, Sep. 2003.

[15] A. Cassidy and A. Andreou, "Dynamical digital silicon neurons," in *Proc. Biomed. Circuits Syst. Conf.*, 2008, pp. 289–292.

[16] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Netw.*, vol. 45, pp. 4–26, 2013.

[17] M. Sivilotti, M. Emerling, and C. Mead, "A novel associative memory implemented using collective computation," in *Proc. Chapel Hill Conf. Very Large Scale Integr.*, 1985, pp. 329–339.

[18] K. A. Boahen, P. O. Pouliquen, A. G. Andreou, and R. E. Jenkins, "A heteroassociative memory using current-mode MOS analog VLSI circuits," *IEEE Trans. Circuits Syst.*, vol. 36, no. 5, pp. 747–755, May 1989.

[19] M. Sivilotti, "Wiring considerations in analog VLSI systems, with application to field-programmable networks," Ph.D. dissertation, Comput. Science Dept., California Inst. Technol., Pasadena, CA, USA, 1991.

[20] M. Mahowald, *An Analog VLSI System for Stereoscopic Vision*. Boston, MA, USA: Kluwer, 1994.

[21] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 416–434, May 2000.

[22] M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, M. Yamada, and A. Masaki, "Design, fabrication and evaluation of a 5-inch wafer-scale neural network LSI composed on 576 digital neurons," in *Proc. Int. Joint Conf. Neural Netw.*, 1990, pp. 527–535.

[23] D. Hammerstrom, "A VLSI architecture for high-performance, low-cost, on-chip learning," in *Proc. Int. Joint Conf. Neural Netw.*, 1990, pp. 537–544.

[24] J. G. Elias, "Artificial dendritic trees," *Neural Comput.*, vol. 5, no. 4, pp. 648–664, 1993.

[25] S. R. Deiss, R. J. Douglas, and A. M. Whatley, "A pulse-coded communications infrastructure for neuromorphic systems," *Pulsed Neural Networks*, W. Maass and C. M. Bishop, Eds. Cambridge, MA, USA: MIT Press, 1999, ch. 6, pp. 157–178.

[26] R. Vogelstein, U. Mallik, J. Vogelstein, and G. Cauwenberghs, "Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 253–265, Jan. 2007.

[27] J. Lin, P. Merolla, J. Arthur, and K. Boahen, "Programmable connections in neuromorphic grids," in *Proc. IEEE Midwest Symp. Circuits Syst.*, 2006, pp. 80–84.

[28] L. Camunas-Mesa, C. Zamarreno-Ramos, A. Linares-Barranco, A. J. Acosta-Jimenez, T. Serrano-Gotarredona, and B. Linares-Barranco, "An event-driven multi-kernel convolution processor module for event-driven vision sensors," *IEEE J. Solid-State Circuits*, vol. 47, no. 2, pp. 504–517, Feb. 2012.

[29] S. Moradi and G. Indiveri, "An event-based neural network architecture with an asynchronous programmable synaptic memory," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 1, pp. 98–107, Feb. 2014.

[30] R. Z. Shi and T. K. Horiuchi, "A summating, exponentially-decaying CMOS synapse for spiking neural systems," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA, USA: MIT Press, 2004.

[31] J. V. Arthur and K. A. Boahen, "Synchrony in silicon: The gamma rhythm," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1815–1825, Nov. 2007.

[32] B. V. Benjamin, J. V. Arthur, P. Gao, P. Merolla, and K. Boahen, "A superposable silicon synapse with programmable reversal potential," in *Proc. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2012, pp. 771–774.

[33] D. R. Frey, "Log-domain filtering: An approach to current-mode filtering," *Inst. Electr. Eng. Proc. G, Circuits Devices Syst.*, vol. 140, no. 6, pp. 406–416, 1993.

[34] P. Merolla and K. A. Boahen, "A recurrent model of orientation maps with simple and complex cells," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA, USA: MIT Press, 2004, pp. 995–1002.

[35] T. Choi, P. Merolla, J. Arthur, K. Boahen, and B. Shi, "Neuromorphic implementation of orientation hypercolumns," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 3, pp. 1049–1060, Jun. 2005.

[36] A. G. Andreou and K. A. Boahen, "Translinear circuits in subthreshold MOS," *Analog Integr. Circuits Signal Process.*, vol. 9, no. 2, pp. 141–166, 1996.

[37] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Comput.*, vol. 10, no. 7, pp. 1601–1638, 1998.

[38] C. Mead and L. Conway, *Introduction to VLSI Systems*. Reading, MA, USA: Addison-Wesley, 1980.

[39] C. Zamarreno-Ramos, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco, "Multicasting mesh AER: A scalable assembly approach for reconfigurable neuromorphic structured AER systems. Application to ConvNets," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 1, pp. 82–102, Feb. 2012.

[40] S. Joshi, S. Deiss, M. Arnold, J. Park, T. Yu, and G. Cauwenberghs, "Scalable event routing in hierarchical neural array architecture with global synaptic connectivity," in *Proc. Int. Workshop Cellular Nanoscale Netw. Appl.*, Feb. 2010, DOI: 10.1109/CNNA.2010.5430296.

[41] S. Scholze, S. Schiefer, J. Partzsch, S. Hartmann, C. Mayr, S. Höppner, H. Eisenreich, S. Henker, B. Vogginger, and R. Schüffny, "VLSI implementation of a 2.8 Gevent/s packet based AER interface with routing and event sorting functionality," *Front. Neurosci.*, vol. 5, no. 117, 2011, DOI: 10.3389/fnins.2011.00117.

[42] P. Merolla, J. V. Arthur, R. Alvarez-Icaza, J.-M. Bussat, and K. Boahen, "A multicast tree router for multichip neuromorphic systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 3, pp. 820–833, Mar. 2014.

[43] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann, 2004.

[44] R. Boppana, S. Chalasani, and C. Raghavendra, "On multicast wormhole routing in multicomputer networks," in *Proc. IEEE Symp. Parallel Distrib. Process.*, 1994, pp. 722–729.

[45] A. Despain and D. Patterson, "X-tree: A structured multiprocessor computer architecture," in *Proc. IEEE Symp. Comput. Architecture*, 1978, pp. 144–151.

[46] S. Browning, "The tree machine: A highly concurrent computing environment," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. Caltech-CS-TR-80-3760, 1980.

[47] E. Horowitz and A. Zorat, "The binary tree as an interconnection network: Applications to multiprocessor systems and VLSI," *IEEE Trans. Comput.*, vol. C-30, no. 4, pp. 247–253, Apr. 1981.

[48] D. Vainbrand and R. Ginosar, "Scalable network-on-chip architecture for configurable neural networks," *Microprocess. Microsyst.*, vol. 35, no. 2, pp. 152–166, 2011.

[49] G. B. Ermentrout and N. Kopell, "Parabolic bursting in an excitable system coupled with a slow oscillation," *SIAM J. Appl. Math.*, vol. 46, no. 2, pp. 233–253, Apr. 1986.

[50] A. Destexhe, Z. F. Mainen, and T. J. Sejnowski, "An efficient method for computing synaptic conductances based on a kinetic model of receptor binding," *Neural Comput.*, vol. 6, no. 1, pp. 14–18, Jan. 1994.

[51] D. I. Feinstein, "The hexagonal resistive network and the circular approximation," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CaltechCSTR:1988.cs-tr-88-07, 1988.

[52] E. Vittoz and J. Fellrath, "CMOS analog integrated circuits based on weak inversion operation," *IEEE J. Solid-State Circuits*, vol. 12, no. 3, pp. 224–231, Jun. 1977.

[53] T. Delbrück and A. V. Schaik, "Bias current generators with wide dynamic range," *Analog Integr. Circuits Signal Process.*, vol. 43, no. 3, pp. 247–268, Jun. 2005.

[54] P. Gao, B. V. Benjamin, and K. Boahen, "Dynamical system guided mapping of quantitative neuronal models onto neuromorphic hardware," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 10, pp. 2383–2394, Oct. 2012.

[55] J. V. Arthur and K. Boahen, "Silicon-neuron design: A dynamical systems approach," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 5, pp. 1034–1043, May 2011.

[56] K. M. Hynna and K. Boahen, "Thermodynamically equivalent silicon models of voltage-dependent ion channels," *Neural Comput.*, vol. 19, no. 2, pp. 327–350, 2007.

[57] K. A. Zaghloul and K. A. Boahen, "An on-off log domain circuit that recreates adaptive filtering in the retina," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 1, pp. 99–107, Jan. 2005.

[58] A. J. Annema, B. Nauta, R. van Langevelde, and H. Tuinhout, "Analog circuits in ultra-deep-submicron CMOS," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 132–143, Jan. 2005.

[59] J. Lin and K. Boahen, "A delay-insensitive address-event link," in *Proc. IEEE Int. Symp. Asynchron. Circuits Syst. (ASYNC)*, 2009, pp. 55–62.

[60] A. Martin, "Programming in VLSI: From communicating processes to delay-insensitive circuits," in *Developments in Concurrency and Communication*. Reading, MA, USA: Addison-Wesley, 1991, pp. 1–64.

[61] A. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proc. IEEE*, vol. 94, no. 6, pp. 1089–1120, Jun. 2006.

[62] K. A. Boahen, "A burst-mode word-serial address-event link I: Transmitter design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1269–1280, Jul. 2004.

[63] K. A. Boahen, "A burst-mode word-serial address-event link II: Receiver design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1281–1291, Jul. 2004.

[64] K. A. Boahen, "A burst-mode word-serial address-event link III: Analysis and test results," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1292–1300, Jul. 2004.

[65] C. Mead and T. Delbruck, "Scanners for visualizing activity of analog VLSI circuitry," *Analog Integr. Circuits Signal Process.*, vol. 1, no. 2, pp. 93–106, 1991.

[66] N. Imam and R. Manohar, "Address-event communication using token-ring mutual exclusion," in *Proc. IEEE Int. Symp. Asynchron. Circuits Syst.*, 2011, pp. 99–108.

[67] A. Chandrasekaran and K. Boahen, "A 1-change-in-4 delay-insensitive interchip link," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 3216–3219.

[68] P. B. McGee, M. Y. Agyekum, M. A. Mohamed, and S. M. Nowick, "A level-encoded transition signaling protocol for high-throughput asynchronous global communication," in *Proc. IEEE Int. Symp. Asynchron. Circuits Syst.*, 2008, pp. 116–127.

[69] S. Choudhary, S. Sloan, S. Fok, A. Neckar, E. Trautmann, P. Gao, T. Stewart, C. Eliasmith, and K. Boahen, "Silicon neurons that compute," in *Proc. Int. Conf. Artif. Neural Netw.*, 2012, pp. 121–128.

[70] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.

[71] J. Schemmel, private communication, 2013.

[72] J. V. Arthur and K. Boahen, "Learning in silicon: Timing is everything," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA, USA: MIT Press, 2006, pp. 75–82.

[73] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, "A learning-enabled neuron array IC based

upon transistor channel models of biological phenomena," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 1, pp. 71–81, Feb. 2013.

[74] D. H. Goldberg, G. Cauwenberghs, and A. G. Andreou, "Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons," *Neural Netw.*, vol. 14, no. 6, pp. 781–793, 2001.

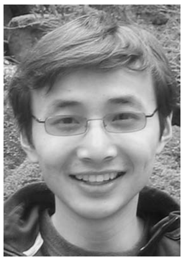
[75] R. J. Vogelstein, F. Tenore, R. Philipp, M. S. Adlerstein, D. H. Goldberg, and G. Cauwenberghs, "Spike timing-dependent plasticity in the address domain," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA, USA: MIT Press, 2002, pp. 1147–1154.

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

ABOUT THE AUTHORS

Ben Varkey Benjamin received the B.Tech. degree in electronics and communication engineering from Mahatma Gandhi University, Kerala, India, in 2005 and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2010, where he is currently working toward the Ph.D. degree in electrical engineering.

He worked for three years as a Design Engineer in the VSBU group at Wipro Technologies, India, where he earned the Prodigy award for the best incoming employee of the year. While there, he also received two U.S. patents for his work on standard cell characterization. He led the testing and characterization of Neurogrid, as well as the design and implementation of the software driver used to program and communicate with Neurogrid. His research focuses on challenges in designing neuromorphic hardware for deep submicrometer technologies.



Peiran Gao received the B.A. degree in neurobiology and physics with minor in electrical engineering and computer science from the University of California Berkeley, Berkeley, CA, USA, in 2009 and the M.S. degree in bioengineering from Stanford University, Stanford, CA, USA, in 2011, where he is currently working toward the Ph.D. degree in bioengineering.

As a Teaching Assistant at Stanford University for the BIOE 332 Large-Scale Neural Modeling course in Spring 2011 Quarter, he developed the dynamical system guided mapping procedure that Neurogrid uses. His research focuses on the theoretical development of a spike-based computational framework.

Emmett McQuinn received the B.S. degree in computer science from Clemson University, Clemson, SC, USA, in 2008, and the M.S. degree in computer science from the University of California San Diego, La Jolla, CA, USA, in 2010.

He was a Research Staff Programmer at Stanford University, Stanford, CA, USA, where he led the development of real-time visualization software for Neurogrid. He then joined the Almaden Research Center, IBM, San Jose, CA, USA, to work on the SyNAPSE project. He is currently working for a startup. His research interests include interactive real-time visualization, hardware-accelerated scientific computing, and scalable neuromorphic systems.

Swadesh Choudhary received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology, Bombay, India, in 2010 and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2012.

He also worked as a Research Assistant at Stanford University to develop a daughterboard for Neurogrid. He is currently a Design Engineer at Intel Corporation, Santa Clara, CA, USA, working in the server development group.

Anand R. Chandrasekaran received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 2001 and the Ph.D. degree in neuroscience from Baylor College of Medicine, Houston, TX, USA, in 2007.

He was a Postdoctoral Scholar in bioengineering at Stanford University, Stanford, CA, USA, working on the Neurogrid project. He is currently the CEO of Mad Street Den, an artificial intelligence company in Bangalore, India.

Jean-Marie Bussat (Member, IEEE) was born in Annecy, France. He received the M.Sc. degree in electrical engineering from ESIGLEC, Rouen, France, in 1995 and the Ph.D. degree in electrical engineering from the University of Paris XI, Orsay, France, in 1998.

He joined the technical staff of the Department of Physics, Princeton University, Princeton, NJ, USA, in 1998, to work on the readout of the electromagnetic calorimeter of the Compact Muon Solenoid (CMS) experiment at CERN, Geneva, Switzerland. He joined the Engineering Division of the Lawrence Berkeley National Laboratory, Berkeley, CA, USA, in 2001 to design instrumentation for physics and material science experiments. In 2007, he joined the Brains in Silicon Laboratory, Stanford University, Stanford, CA, USA, to work on Neurogrid.

Rodrigo Alvarez-Icaza received the B.S. degree in mechanical and electrical engineering from the Universidad Iberoamericana, Mexico City, Mexico, in 1999, the M.S. degree in bioengineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2005, and the Ph.D. degree in bioengineering from Stanford University, Stanford, CA, USA, in 2010.

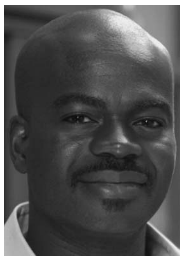
He is currently a research staff member at the Almaden Research Center, IBM, San Jose, CA, USA, where his research focuses on brain-inspired computers.

John V. Arthur received the B.S.E. degree (*summa cum laude*) in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2000 and the Ph.D. degree in bioengineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2006.

He was a Postdoctoral Scholar in bioengineering at Stanford University, Stanford, CA, USA, as a lead on the Neurogrid project. He is currently a research staff member at the Almaden Research Center, IBM, San Jose, CA, USA, working on the SyNAPSE project. His research interests include dynamical systems, neuromorphic and neurosynaptic architecture, and hardware-aware algorithm design.

Paul A. Merolla received the B.S. degree in electrical engineering (with high distinction) from the University of Virginia, Charlottesville, VA, USA, in 2000 and the Ph.D. degree in bioengineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2006.

His research is to build more intelligent computers, drawing inspiration from neuroscience, neural networks, and machine learning. He was a Postdoctoral Scholar in the Brains in Silicon Laboratory, Stanford University, Stanford, CA, USA (2006–2009), working as a lead chip designer on Neurogrid, an affordable supercomputer for neuroscientists. Starting in 2010, he has been a research staff member at the Almaden Research Center, IBM, San Jose, CA, USA, where he was a lead chip designer for the first fully digital neurosynaptic core as part of the DARPA-funded SyNAPSE project. His research includes low-power neuromorphic systems, asynchronous circuit design, large-scale modeling of cortical networks, statistical mechanics, machine learning, and probabilistic computing.



Kwabena Boahen (Senior Member, IEEE) received the B.S. and M.S.E. degrees in electrical and computer engineering from The Johns Hopkins University, Baltimore, MD, USA, both in 1989 and the Ph.D. degree in computation and neural systems from the California Institute of Technology, Pasadena, CA, USA, in 1997.

He was on the bioengineering faculty of the University of Pennsylvania, Philadelphia, PA, USA, from 1997 to 2005, where he held the first Skirkanich Term Junior Chair. He is currently an Associate Professor in the Bioengineering Department, Stanford University, Stanford, CA, USA. He directs the Brains in Silicon Laboratory, Stanford University, which develops silicon-integrated circuits that emulate the way neurons compute, linking the seemingly disparate fields of electronics and computer science with neurobiology and medicine. His contributions to the field of neuromorphic engineering include a silicon retina that could be used to give the blind sight, a self-organizing chip that emulates the way the developing brain wires itself up, and a specialized hardware platform (Neurogrid) that simulates a million cortical neurons in real time rivaling a supercomputer while consuming only a few watts.

Dr. Boahen received several distinguished honors, including a Fellowship from the Packard Foundation (1999), a CAREER award from the National Science Foundation (2001), a Young Investigator Award from the U.S. Office of Naval Research (2002), and the National Institutes of Health Director's Pioneer Award (2006) and Transformative Research Award (2011). His scholarship is widely recognized, with over 80 publications to his name, including a cover story in the May 2005 issue of *Scientific American*.