Cognitive Computing Building Block: A Versatile and Efficient Digital Neuron Model for Neurosynaptic Cores

Andrew S. Cassidy, Paul Merolla, John V. Arthur, Steve K. Esser, Bryan Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada[†], Theodore M. Wong, Vitaly Feldman, Arnon Amir, Daniel Ben-Dayan Rubin[§], Filipp Akopyan, Emmett McQuinn, William P. Risk, and Dharmendra S. Modha
IBM Research - Almaden, San Jose, CA 95120 [†]IBM Research - Austin, Austin, TX 78758

***Abstract*—Marching along the DARPA SyNAPSE roadmap, IBM unveils a trilogy of innovations towards the TrueNorth cognitive computing system inspired by the brain's function and efficiency. Judiciously balancing the dual objectives of functional capability and implementation/operational cost, we develop a simple, digital, reconfigurable, versatile spiking neuron model that supports one-to-one equivalence between hardware and simulation and is implementable using only 1272 ASIC gates. Starting with the classic leaky integrate-and-fire neuron, we add: (a) configurable and reproducible stochasticity to the input, the state, and the output; (b) four leak modes that bias the internal state dynamics; (c) deterministic and stochastic thresholds; and (d) six reset modes for rich finite-state behavior. The model supports a wide variety of computational functions and neural codes. We capture 50+ neuron behaviors in a library for hierarchical composition of complex computations and behaviors. Although designed with cognitive algorithms and applications in mind, serendipitously, the neuron model can qualitatively replicate the 20 biologically-relevant behaviors of a dynamical neuron model.**

I. Introduction

A. Context

To usher in a new era of cognitive computing [1], we are developing TrueNorth (Fig. 1), a non-von Neumann, modular, parallel, distributed, event-driven, scalable architecture—inspired by the function, low power, and compact volume of the organic brain. TrueNorth is a versatile substrate for integrating spatio-temporal, real-time cognitive algorithms for multi-modal, sub-symbolic, sensor-actuator systems. TrueNorth comprises of a scalable network of configurable neurosynaptic cores. Each core brings memory ("synapses"), processors ("neurons"), and communication ("axons") in close proximity, wherein inter-core communication is carried by all-or-none spike events, sent over a message-passing network.

Recently, we have achieved a number of milestones: first, a demonstration of 256-neuron, 64k/256k-synapse neurosynaptic cores in 45nm silicon [2], [4] that were featured on the cover of *Scientific American* in December 2011; second, a demonstration of multiple real-time applications [5]; third, Compass, a simulator of the TrueNorth architecture, which simulated over 2 billion neurosynaptic cores exceeding 10^{14} synapses [3], [6]; and, fourth, a visualization of the long-distance connectivity of the Macaque brain [7]—mapped to TrueNorth architecture—that was featured on the covers of *Science* [8] and *Communications of the ACM* [1].

[§]Work done while at IBM Research - Almaden.

We unveil a series of interlocking innovations in a set of three papers. In this paper, we present a versatile and efficient digital, spiking neuron model that is a building block of the TrueNorth architecture. In two companion papers, we introduce a programming paradigm for hierarchically composing and configuring cognitive systems, that is effective for the programmer and efficient for the TrueNorth architecture [9] as well as present a set of algorithms and applications that demonstrate the potential of the TrueNorth architecture and value of the programming paradigm [10].

B. Motivation

Biological cells found in the brain, coined "neurons" by Heinrich von Waldeyer-Hartz in the late 19th century, have been modeled at different levels of abstraction over the last hundred years (see Table I for an abbreviated chronology). These models range from *phenomenological*, where the goal is to capture the input-output behavior of a neuron using simple mathematical abstractions (e.g., McCulloch-Pitts [11]), to *biophysical*, where the goal is to model the electrophysiology of neuronal membranes (e.g., Hodgkin-Huxley [12]). In the context of the TrueNorth system architecture, we seek a neuron model that balances the dual objectives of capability (from a computational perspective) and cost (from an implementation perspective), which are both functions of the underlying complexity of the neuron model. Neuron capability should be sufficient to support useful and interesting cognitive algorithms [10], while the cost should be no more than necessary in terms of power, area, and speed.

C. Contribution

As our main contribution, judiciously balancing capability and cost, we follow a digital, mathematical, and phenomenological perspective to develop a simple, reconfigurable, versatile neuron model. This model has defined input-output behavior for a wide range of applications, has heterogeneity and variability in input-output behavior across population and time, consumes few transistors, is amenable to implementation in a dense CMOS process, and can be reproducibly and predictably simulated for one-to-one equivalence between hardware and simulation. We summarize the construction, the capability, the composability, and the costs of the proposed neuron model.

Construction: In Section II, starting with the classic leaky integrate-and-fire neuron, we add configurable and reproducible stochasticity to the input, the state, and the output of the model. Specifically, we introduce stochastic synaptic input, leak, and threshold, enabling rich dynamics across population

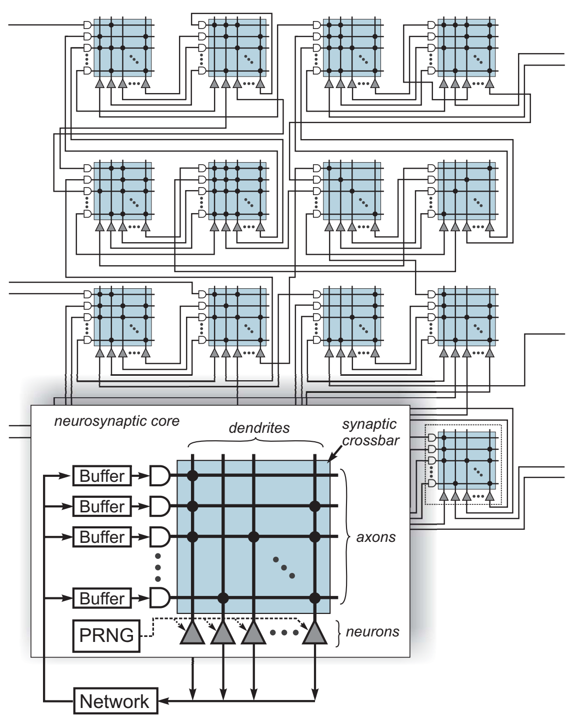

Fig. 1. TrueNorth is a brain-inspired chip architecture built from an interconnected network of lightweight neurosynaptic cores [2], [3]. TrueNorth implements "gray matter" short-range connections with an intra-core crossbar memory and "white matter" long-range connections through an inter-core spike-based message-passing network. TrueNorth is fully programmable in terms of both the "physiology" and "anatomy" of the chip, that is, neuron parameters, synaptic crossbar, and inter-core neuron-axon connectivity allow for a wide range of structures, dynamics, and behaviors. *Inset:* The TrueNorth neurosynaptic core has 256 axons, a 256×256 synapse crossbar, and 256 neurons. Information flows from axons to neurons gated by binary synapses, where each axon fans out, in parallel, to all neurons thus achieving a 256-fold reduction in communication volume compared to a point-to-point approach. A conceptual description of the core's operation follows. To support multi-valued synapses, axons are assigned types which index a synaptic weight for each neuron. Network operation is governed by a discrete time step. In a time step, if the synapse value for a particular axon-neuron pair is non-zero and the axon is active, then the neuron updates its state by the synaptic weight corresponding to the axon type. Next, each neuron applies a leak, and any neuron whose state exceeds its threshold fires a spike. Within a core, *PRNG* (pseudorandom number generator) can add noise to the spike thresholds and stochastically gate synaptic and leak updates for probabilistic computation; *Buffer* holds incoming spikes for delayed delivery; and *Network* sends spikes from neurons to axons.

and time. We then introduce four leak modes that bias the internal state dynamics so that neurons can have radically different responses to identical inputs. Specifically, we allow for leaks that subtract from or add to the membrane potential as well as allow the membrane potential to diverge away from or converge towards a resting potential. Furthermore, we introduce two threshold modes that allow a deterministic or a stochastic threshold, so that neurons can fire differently even with the same accumulated membrane potential. Finally, we introduce six reset modes that determine the value of the membrane potential after firing, enabling a rich finite-state transition behavior.

Capability and Composition: In Section III, by exploiting the parametric approach of our neuron model, we demonstrate a wide variety of computational functions; for example, arithmetic, control, data generation, logic, memory, classic neuron behaviors, signal processing, and probabilistic computation. The model can support a variety of neural codes including rate, population, binary, and time-to-spike, thus permitting a rich language for inter-neuron communication. To make it easy to use the neurons, we have created a parametrized and characterized neuron function library, with 50+ elements, containing fundamental building blocks. Drawing inspiration from RISC philosophy, by composing multiple neurons together we can synthesize an extremely rich and diverse array of complex computations and behaviors from simpler library elements. While fidelity with neuroscientifically observed neuron behaviors was not our express design goal, we show in Section IV, quite surprisingly, that we were able to qualitatively replicate the 20 behaviors of the Izhikevich dynamical neuron model [24] using a small number of elementary neurons.

Cost: By design, our neuron model uses only simple addition and multiplexing arithmetic/logic units, avoiding complex function units such as multiplication, division, and exponentiation. It can be implemented using only fixed-point arithmetic, avoiding complex floating-point circuitry. As a result, when mapped to an ASIC standard cell library for fabrication in a state-of-the-art silicon process, the neuron model is implementable using only 1272 gates (924 gates for the model computation and 348 gates for the random number generator). From an operational perspective, addition and multiplexing circuits are intrinsically lower power than more complex arithmetic function circuits. It is noteworthy that the relatively slow firing frequency of neurons operating in real-time, as compared to the speed of modern silicon, allows us the possibility to reduce cost in three ways. First, we can reuse physical arithmetic circuits, reducing the aggregate implementation area drastically. Second, we can power-gate, turning off power to these circuits while they are quiescent, reducing the total power consumption. Third, the neuron can be implemented in an event-driven fashion so that its active power consumption is based on the number of spike events it has to process (for an example, see [29]).

TABLE I. A REPRESENTATIVE COMPENDIUM OF NEURON MODELS.

Year	Model Name	Reference
1907	Integrate and fire	[13]
1943	McCulloch and Pitts	[11]
1952	Hodgkin-Huxley	[12]
1958	Perceptron	[14]
1961	Fitzhugh-Nagumo	[15]
1965	Leaky integrate-and-fire	[16]
1981	Morris-Lecar	[17]
1986	Quadratic integrate-and-fire	[18]
1989	Hindmarsh-Rose	[19]
1998	Time-varying integrate-and-fire model	[20]
1999	Wilson Polynomial	[21]
2000	Integrate-and-fire or burst	[22]
2001	Resonate-and-fire	[23]
2003	Izhikevich	[24]
2003	Exponential integrate-and-fire	[25]
2004	Generalized integrate-and-fire	[26]
2005	Adaptive exponential integrate-and-fire	[27]
2009	Mihalas-Neibur	[28]
2013	This work	—

SYNAPTIC INTEGRATION

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{N-1} x_i(t)\ s_i \quad (1)$$

LEAK INTEGRATION

$$V_j(t) = V_j(t) - \lambda_j \quad (2)$$

THRESHOLD, FIRE, RESET

$$\begin{array}{lll} \text{if} & V_j(t) \geq \alpha_j & (3) \\ & \text{Spike} & (4) \\ & V_j(t) = R_j & (5) \\ \text{endif} & & (6) \end{array}$$

Fig. 2. Leaky Integrate and Fire Neuron Equations.

II. Neuron Specification

Our neuron model is based on the leaky integrate-and-fire neural model with a constant leak, which we augmented in several ways. We begin by briefly reviewing the classic leaky integrate-and-fire neural model, followed by an in-depth description of our neuron model.

A. The Leaky Integrate and Fire (LIF) Neuron

The operation of the leaky integrate-and-fire (LIF) neuron model with a constant leak is described by five basic operations: 1. synaptic integration, 2. leak integration, 3. threshold, 4. spike firing, and 5. reset. The LIF neuron model is summarized in the general case in Eqns. (1)-(6) in Fig. 2. For the j^{th} neuron in the t^{th} timestep, the membrane potential $V_j(t)$ is the sum of the membrane potential in the previous timestep $V_j(t-1)$ and the synaptic input. For each of the N synapses, the synaptic input is the sum of the spike input to the synapse $x_i(t)$ at the current timestep, multiplied by the signed synaptic weight s_i. Following integration, the LIF neuron model subtracts the leak value λ_j from the membrane potential. With a linear leak, this constant is subtracted every timestep, regardless of membrane potential or synaptic activity. This operation serves as a constant bias on the neural dynamics. Then, the LIF neuron model compares the membrane potential at the current timestep $V_j(t)$ with the neuron threshold α_j. If the membrane potential is greater than or equal to the threshold voltage, the neuron fires a spike and resets its membrane potential. In the typical case, the reset voltage R_j is zero. This basic model of neural computation, graphically depicted in Fig. 3, can be used to generate a wide variety of functions and behaviors.

B. The Full Neuron Model

The complete specification of our neuron model[1] is given in Eqns. (10)-(19) in Fig. 4. The symbols are summarized in

[1] For simplicity, we have omitted hardware-centric implementation details relevant to the neuron equation, including 1. fixed-point arithmetic, 2. floor and ceiling checks to prevent arithmetic overflow, and 3. a check in convergent leak mode that prevents ringing about zero for $V_j(t)$. We also use an additional ceiling check in non-reset ($\gamma_j = 2$) mode, to prevent $V_j(t)$ from going out of range of the stochastic threshold.

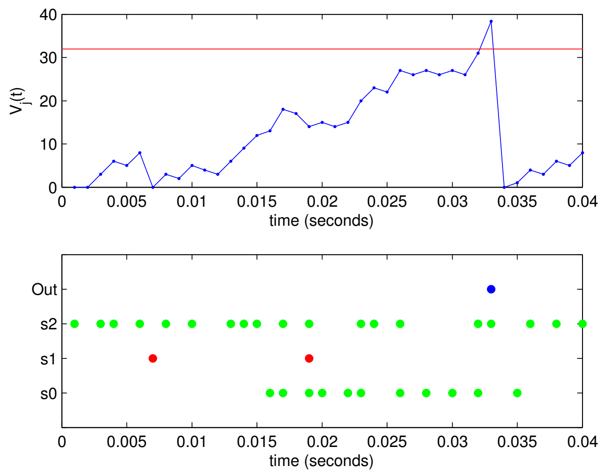

Fig. 3. Integrate and fire neuron. Top: membrane potential $V_j(t)$ and firing threshold $\alpha_j = 32$. Bottom: spike raster plot (excitatory spike input s_0, s_2, inhibitory spike input s_1, spike output), where each dot represents a spike event in time.

Table II. The symbols in the bottom half of the table are user-configurable parameters, while the symbols in the upper half of the table are system variables. The specification uses the subfunctions signum:

$$\text{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0 \end{cases} \quad (7)$$

the binary comparison operation for conditional stochastic evaluation (see Section II-D):

$$F(s, \rho) = \begin{cases} 1 & \text{if } |s| \geq \rho \\ 0 & \text{else} \end{cases} \quad (8)$$

the binary bitwise AND operation: &, and the Kronecker delta function:

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{else} \end{cases} \quad (9)$$

The details of the neuron operation are covered in the following subsections.

C. Synaptic Integration: Crossbar

The neurosynaptic core includes a synaptic crossbar, connecting axons and neurons (Fig. 1). Incoming spikes arrive, targeting an axon, and then hit all of the active synapses on the axon. The synaptic integration Eqn. (1) becomes Eqn. (10), where $A_i(t)$ is activity on the i^{th} axon at time t and $w_{i,j}$ is the synaptic crossbar matrix. The axon activity $A_i(t)$ is 1 if there is a spike present at the current (t^{th}) timestep, and 0 otherwise. The entries in the synaptic crossbar matrix $w_{i,j}$ are 1 if there is a synaptic connection between the i^{th} axon and the j^{th} neuron and 0 otherwise.

To enable multi-value synapses while using a binary synaptic crossbar, we assign a type G_i to each axon. Then each neuron has an individual signed integer weight for that axon type. In the current instantiation, there are four possible axon types $\{G_i \in 0, 1, 2, 3\}$, and each neuron has four signed

SYNAPTIC INTEGRATION

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{255} A_i(t)\ w_{i,j} \left[(1 - b_j^{G_i}) s_j^{G_i} + b_j^{G_i} F(s_j^{G_i}, \rho_{i,j}) \mathrm{sgn}(s_j^{G_i}) \right] \quad (10)$$

LEAK INTEGRATION

$$\Omega = (1 - \epsilon_j) + \epsilon_j\ \mathrm{sgn}(V_j(t)) \quad (11)$$

$$V_j(t) = V_j(t) + \Omega\ [(1 - c_j^{\lambda})\lambda_j + c_j^{\lambda} F(\lambda_j, \rho_j^{\lambda})\ \mathrm{sgn}(\lambda_j)] \quad (12)$$

THRESHOLD, FIRE, RESET

$$\eta_j = \rho_j^T\ \&\ M_j \quad (13)$$

$$\text{if} \quad V_j(t) \geq \alpha_j + \eta_j \quad (14)$$

Spike (15)

$$V_j(t) = \delta(\gamma_j) R_j + \delta(\gamma_j - 1)(V_j(t) - (\alpha_j + \eta_j)) + \delta(\gamma_j - 2) V_j(t) \quad (16)$$

$$\text{elseif} \quad V_j(t) < -[\beta_j \kappa_j + (\beta_j + \eta_j)(1 - \kappa_j)] \quad (17)$$

$$V_j(t) = -\beta_j \kappa_j + [-\delta(\gamma_j) R_j + \delta(\gamma_j - 1)(V_j(t) + (\beta_j + \eta_j)) + \delta(\gamma_j - 2) V_j(t)] (1 - \kappa_j) \quad (18)$$

endif (19)

Fig. 4. Neuron Specification Equations.

weights $s_j^{G_i}$ associated with the four axon types G_i. When a spike arrives on any axon of the G_i^{th} type, a neuron with a synapse on that axon will integrate its signed weight of the G_i^{th} type $s_j^{G_i}$, resulting in the synaptic integration summation: $\sum_{i=0}^{255} A_i(t)\ w_{i,j}\ s_j^{G_i}$. (This equation assumes that the synapse configuration bit for the axon type is set to deterministic mode, $b_j^{G_i} = 0$.)

D. Stochastic Synaptic and Leak Integration

For each neuron, each synaptic weight and leak has a configuration bit, $b_j^{G_i}$ and c_j^{λ} respectively, where setting the bit to 0 selects deterministic mode, and 1 selects stochastic mode. For stochastic synaptic and leak integration, operation is as follows. Every time a valid synaptic or leak event occurs, the neuron draws a uniformly distributed random number ρ_j. If the synaptic weight $s_j^{G_i}$ or leak weight λ_j is greater than or equal to the drawn random number ρ_j, then the neuron integrates $\{-1, +1\}$ otherwise, it does not integrate. This behavior is described mathematically as follows. The value integrated $\{-1, +1\}$, is determined by the sign of the respective synapse: $\mathrm{sgn}(s_j^{G_i})$ or leak: $\mathrm{sgn}(\lambda_j)$. Stochastic operation is represented using the binary comparison operator $F(s, \rho)$, where s is a scalar value, and ρ is a random number. If the scalar value is greater than or equal to the random number, $F(s, \rho)$ returns 1, otherwise it returns 0, according to Eqn. (8). Combining deterministic mode and stochastic mode, the full equation for synaptic integration is given in Eqn. (10) and leak integration in Eqn. 12).

TABLE II. SUMMARY OF SYMBOLS. SYSTEM VARIABLES IN UPPER SECTION. USER CONFIGURABLE PARAMETERS IN LOWER SECTION. PSEUDO-RANDOM NUMBER IS ABBREVIATED PRN.

Variables and Parameters	Symbol	Format
membrane potential	$V_j(t)$	signed int
local timestep	t	unsigned int
input spikes on i^{th} axon	$A_i(t)$	$\{0, 1\}$
synaptic PRN	$\rho_{i,j}$	unsigned int
leak PRN	ρ_j^{λ}	unsigned int
threshold PRN (drawn)	ρ_j^T	unsigned int
threshold PRN (masked)	η_j	unsigned int
leak direction variable	Ω	$\{-1, 0, +1\}$
synapse (i^{th} axon, j^{th} neuron)	$w_{i,j}$	$\{0, 1\}$
type of i^{th} axon	G_i	$\{0, 1, 2, 3\}$
synaptic weight/probability	$s_j^{G_i}$	signed int
synaptic weight/probability select	$b_j^{G_i}$	$\{0, 1\}$
leak-reversal flag	ϵ_j	$\{0, 1\}$
leak weight/probability	λ_j	signed int
leak weight/probability select	c_j^{λ}	$\{0, 1\}$
positive $V_j(t)$ threshold	α_j	unsigned int
negative $V_j(t)$ threshold/floor	β_j	unsigned int
threshold PRN mask	M_j	unsigned int
reset voltage	R_j	signed int
negative thresh: reset or saturate	κ_j	$\{0, 1\}$
$V_j(t)$ reset mode	γ_j	$\{0, 1, 2\}$
PRNG initial seed value	ρ_j^{seed}	unsigned int

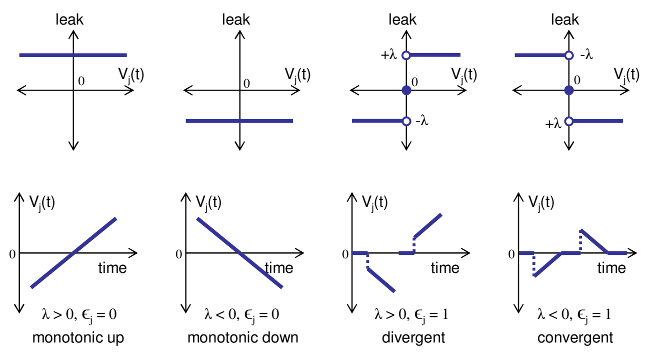

Fig. 5. Four leak modes. The upper plots show the leak, depending on the membrane potential. The lower plots show the time course of the membrane potential given the leak. The divergent and convergent leaks require synaptic input (dashed lines) to move the membrane potential away from zero.

E. Leak Modes

We augment the linear leak so that it can be positive or negative, and we introduce a “leak-reversal” mode. With standard leak operation, the signed leak is always directly integrated, regardless of the value of the membrane potential. The leak is always positive or negative, leading to a bias that monotonically increases or decreases the membrane potential. In leak-reversal mode, the signed leak is directly integrated when the membrane potential is above zero, and the sign is reversed when the membrane potential is below zero. (At zero, the membrane potential does not leak, remaining zero.) This

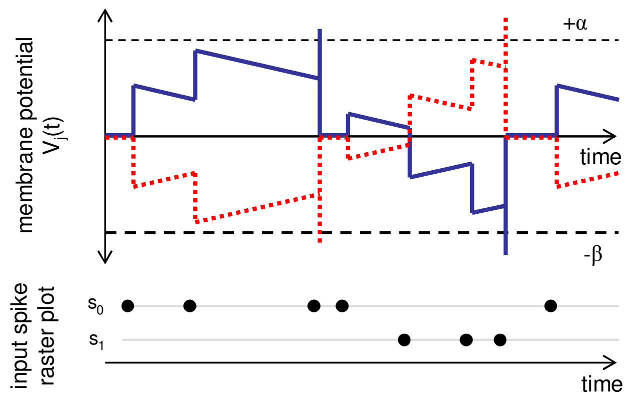

Fig. 6. ON-OFF neuron pairs. The membrane potential for the two neurons are shown in red and blue respectively. The reset voltage R_j is zero for both neurons and $\kappa_j = 1$. Incident spikes on s_0 are excitatory for the blue neuron, but inhibitory for its symmetric pair (in red). Similarly, incident spikes on s_1 are inhibitory for the blue neuron, but excitatory for its symmetric pair.

mechanism creates two leak-reversal modes: a convergent leak, where the neuron membrane potential leaks toward zero, from both above ($V_j > 0$) and below ($V_j < 0$), as well as a divergent leak, where the neuron membrane potential leaks away from zero, toward the positive threshold above zero ($+\alpha_j$) and toward the negative threshold below zero ($-\beta_j$). These leak modes are summarized in Fig. 5. We select between the modes using the leak-reversal parameter, ϵ_j. The leak also supports a stochastic mode as described in the previous section II-D. Stochastic mode may be combined with any one of the four leak modes. Deterministic or stochastic mode is selected using the parameter c_j^λ, resulting in the full equations for leak integration (11) and (12).

F. Thresholds

In addition to the positive threshold we described for a leaky integrate-and-fire neuron (3), we introduce a negative threshold $-\beta_j$, for when the neuron membrane potential integrates below zero. This negative threshold has two different behaviors when crossed. In the first case, the negative threshold is a floor, so that when the membrane potential V_j crosses $-\beta_j$, V_j stays at $-\beta_j$. In the second case, we define a "bounce" when the membrane potential V_j crosses $-\beta_j$. In this case, V_j is reset to the negative reset value $-R_j$. No output spike is generated when we cross below $-\beta_j$. This feature allows for a pair of neurons to be kept in lock-step with inverted parameters (Fig. 6). For example, we can define an ON-OFF neuron pair, where one neuron fires when an ON stimulus is received and the OFF neuron fires when the stimulus is absent. The negative threshold with "bounce" ensures that the membrane potential of both neurons will be exactly inverted copies of each other. We select the floor behavior by setting $\kappa_j = 1$ and the "bounce" behavior by setting $\kappa_j = 0$, as defined by (17).

G. Stochastic Thresholds

The neuron model supports a random threshold η_j, which is added to the deterministic thresholds α_j and β_j. The random threshold value η_j is formed by the bitwise AND of M_j and the random number generator value ρ_j^T, as given in Eqn. (13). The mask M_j is a ones mask of configurable width, starting at the least significant bit. This mask scales the range of the random number η_j that is added to the threshold. For example, a hexadecimal mask value of 0x000F generates a uniform random number from 0 to 15, while a value of 0x01FF generates a uniform random number from 0 to 511. With a mask of zero, the random value is always zero, effectively disabling the random portion of the threshold value. The positive threshold check with random threshold is Eqn. (14), while the negative threshold check (17) includes the random threshold only in "bounce" mode ($\kappa_j = 0$).

H. Reset Modes

Next we introduce three modes that govern the reset behavior of the neuron: normal mode ($\gamma_j = 0$), linear mode ($\gamma_j = 1$), and non-reset mode ($\gamma_j = 2$). In normal mode ($\gamma_j = 0$), the behavior follows the standard integrate-and-fire model, where the membrane potential V_j is set to the reset voltage R_j after crossing the positive threshold and firing a spike (14). Any residual potential above the threshold is discarded. For example, if $\alpha_j = 100$ and V_j integrates to 110 in a single timestep, the neuron fires, resets to R_j, and the amount above the threshold, 10, is ignored. R_j can be assigned positive, negative, or zero values. In linear mode ($\gamma_j = 1$), the residual potential above the threshold is **not** discarded. For example, if $\alpha_j = 100$ and V_j integrates to 110 in a single timestep, the neuron fires, and resets to the amount above the threshold: $V_j = 10$. R_j is not used in this mode. Non-reset mode ($\gamma_j = 2$), a special case intended to be used with the stochastic threshold (described in section II-G), creates a stochastic neuron type. When crossing the stochastic threshold ($\alpha_j + \eta_j$), the neuron does not reset its membrane potential. In this mode, synaptic or leak integration must be used to pull the membrane potential below the threshold. Using Kronecker delta function notation (9), the positive reset equation is (16) and the negative reset equation is (18).

The configuration modes are summarized in Table III.

III. Neuron Computational Function Library

The primary objective of our neural specification is for building synthetic cognitive applications, motivating a neuron specification that is rich in computational expressiveness. It must not only have complex spiking output patterns to diverse inputs, but we must also be able to harness those complex behaviors to perform useful computation.

In this section, we present a library of functions that are computable using our neuron specification. Most can be computed with a single neuron, while other more complex functions require a few neurons. Without space to present every function in detail, we summarize the function library in Table IV and present a more detailed description of only a few example functions. The function library spans a wide range of different classes of operations, and accommodates a variety of spike codes including rate code, population code, binary code, time-to-spike, and rank code.

A. Rate Store Neuron

As a first example, we present the operation of the stochastic spiking rate store neuron as shown in Fig. 7. This neuron

TABLE III. CONFIGURATION MODE SUMMARY.

$b_j^{G_i}$		Synapse Modes	Integration Value	
0		deterministic weight	$s_j^{G_i}$	
1		stochastic integration	$F(s_j^{G_i}, \rho_{i,j})\text{sgn}(s_j^{G_i})$	
c_j^{λ}		Leak Modes	Integration Value	
0		deterministic leak	λ_j	
1		stochastic leak integration	$F(\lambda_j, \rho_j^{\lambda})\text{sgn}(\lambda_j)$	
ϵ_j	sgn(λ_j)	Leak Direction Modes		
0	+1	monotonic up		
0	-1	monotonic down		
1	+1	divergent, leak to $+/-$ thresholds		
1	-1	convergent, leak to zero		
M_j	κ_j	Threshold Modes	Positive Threshold	Negative Threshold
$= 0$	-	deterministic threshold	α_j	$-\beta_j$
$\neq 0$	0	stochastic threshold ($\eta_j = \rho_j^T$ & M_j)	$\alpha_j + \eta_j$	$-(\beta_j + \eta_j)$
$\neq 0$	1	stochastic threshold ($\eta_j = \rho_j^T$ & M_j)	$\alpha_j + \eta_j$	$-\beta_j$
γ_j	κ_j	$V_j(t)$ Reset Mode	Positive Reset	Negative Reset
0	0	normal	R_j	$-R_j$
0	1	normal - negative saturation	R_j	$-\beta_j$
1	0	linear	$V_j - (\alpha_j + \eta_j)$	$V_j + (\beta_j + \eta_j)$
1	1	linear - negative saturation	$V_j - (\alpha_j + \eta_j)$	$-\beta_j$
2	0	non-reset	V_j	V_j
2	1	non-reset - negative saturation	V_j	$-\beta_j$

TABLE IV. NEURON FUNCTION LIBRARY SUMMARY TABLE.

Type	Name (# of neurons)
Arithmetic	absolute value (3), addition (1), division (2), fixed-gain div./mult. (1), log (1), min (2), max (2), modulation (1), multiplication (1), rate clip (2), rate match (3), sigmoid (1), square root (3), square (2), subtraction (1)
Control	bistable (1), tristable (1), delay (1), one shot (1), pass gate (1)
Data Generation	spontaneous (1), ramp up/down (3), triangle wave (5), random distributions (1)
Logic	AND (1), NAND (1), NOR (1), NOT (1), OR (1), XNOR (3), XOR (3)
Memory	binary (1), rate store (1)
"Neural"	integrate-and-fire (1), bursting (2), coincidence (1), McCulloch-Pitts (1), Boltzmann (1), motion history (1), on/off pair (2) , onset/offset pair (2)
Signal Processing	decorrelation (1), rate attractor (1), convolution (1), low-pass filter (1), high-pass filter (1), band-pass filter (1), spatiotemporal filter (1), Bloom filter (1)
Probabilistic	noisy-OR: prob. union (1), noisy-AND: prob. intersection (1)
Time-to-Spike	L1 distance (2), max (N), min (N), median (N+1), coincidence (1), anti-coincidence (1), matched filter (1)

can act as a multi-valued memory, storing a spike rate. It will stochastically fire spikes at a given rate until its membrane potential is modified. This neuron type uses a non-zero stochastic threshold and non-reset mode ($\gamma_j = 2$). In Fig. 7,

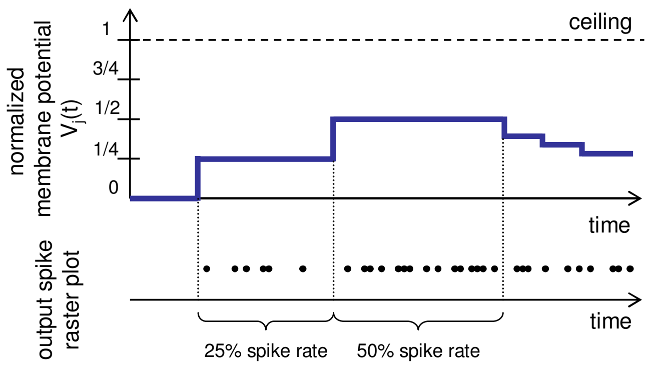

Fig. 7. Rate store neuron fires stochastically in proportion to the value of the membrane potential, $V_j(t)$.

synaptic input increases and decreases the membrane potential. While the membrane potential is non-zero, the neuron will spike stochastically, with probability proportional to the value of the membrane potential, within the range between 0 and the ceiling (set by the random threshold mask). For example, when the membrane potential is 1/4 of the range between 0 and the ceiling, the neuron fires 25% of the time, on average. Similarly, when the membrane potential is 1/2 of the range between 0 and the ceiling, the neuron fires 50% of the time, on average. (In non-reset mode ($\gamma_j = 2$), the ceiling is defined as the sum of the deterministic threshold and the full-range random number value $\alpha_j + M_j$.)

This neuron type efficiently emulates the state holding behavior of populations of recurrent neurons. As such, it is a behavioral abstraction of biological function. Populations of recurrent neurons have been shown to be useful in models of

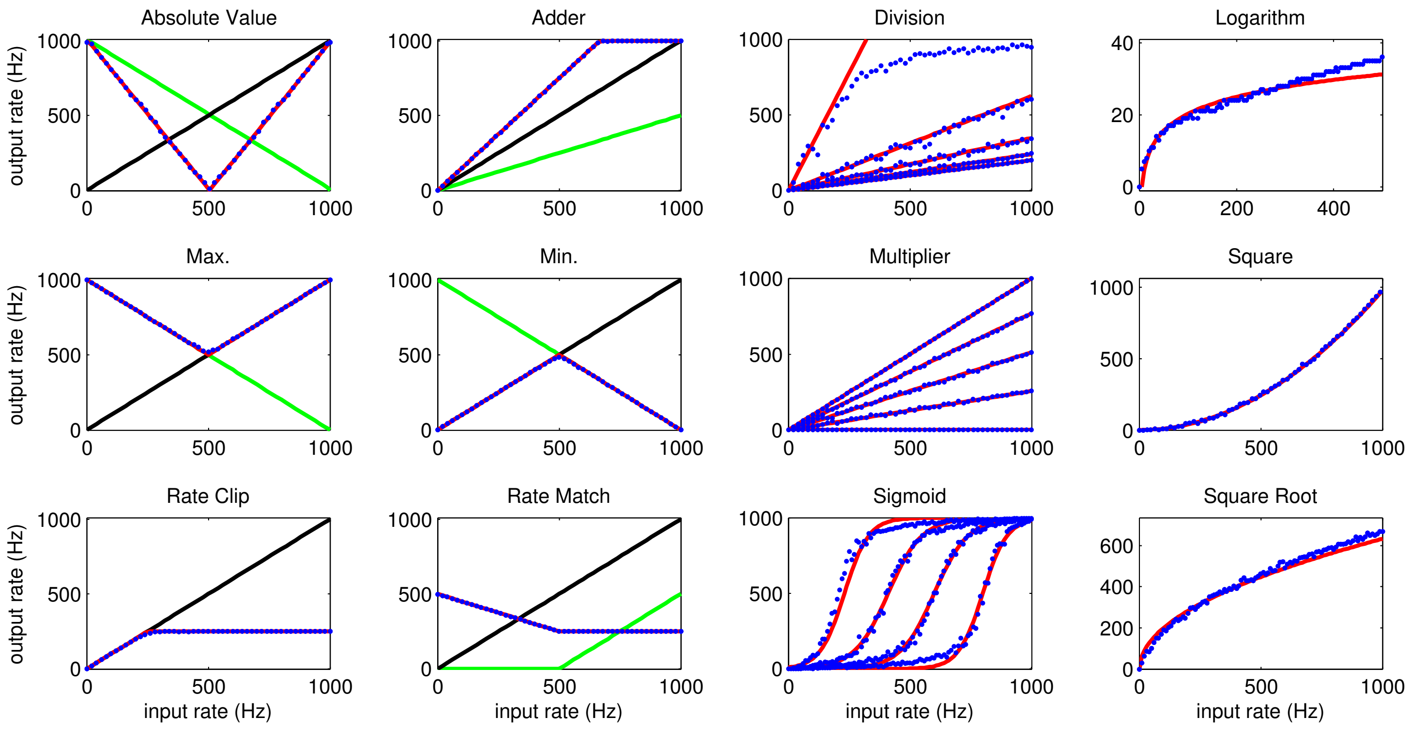

Fig. 8. Arithmetic functions (left to right, top to bottom): absolute value = abs(green-black), adder = (green+black), division, logarithm, maximum = max(green,black), minimum = min(green,black), multiplication, square, rate clip = min(black,250), rate match = 500-abs(green-black)/2, sigmoid, and square root. Actual output response is shown in blue, expected response in red. Green and black points are inputs.

working memory [30], finite state machines [31], and graphical models [32].

B. Arithmetic Functions

Fig. 8 shows the transfer functions of 12 different arithmetic functions. The horizontal axis corresponds to the input rate of one of the function inputs. The vertical axis is the corresponding output rate. One trial generates one data point by presenting a stochastic stimulus with a constant average rate and integrating the output response over 1000 timesteps (1 second). By sweeping one or both inputs over the range (from 0 to 1000 Hz), we generate the curves from many successive trials. In the multiplication and division plots, one input was held constant while sweeping the other input multiple times, to generate each curve in the figure.

IV. Biologically Relevant Neuron Behaviors

In [24] Izhikevich reviewed 20 of the most prominent features of biological spiking neurons. While our primary motivation is to create synthetic computations, we implement these 20 behaviors to demonstrate that our neural model is sufficiently rich to replicate biologically plausible behavior. In recreating these behaviors, we stick to our minimalist approach, creating complex behaviors from more simple building blocks. We replicate eleven behaviors using single neurons, seven more using a pair of neurons, and the final two complex behaviors using only three neurons.

Just as Izhikevich makes qualitative comparisons between biology and his dynamical model, we have qualitatively replicated these 20 behaviors, as shown in Fig. 9. We do not use the same dynamical systems mechanism as Izhikevich to generate the spiking patterns, so the time course of the membrane potentials $V_j(t)$ does not identically match those of [24]. However, using a similar input stimulus, the spiking patterns match, and most importantly, we replicate the qualitative spirit of the behaviors. Similar to the step taken by dynamical system models to abstract from the electro-physiology of ion channels to model the functional neuron dynamics, we abstract from dynamical system models to replicate functional neuron behavior (i.e. spike patterns). Similar success in duplicating neuron spike behavior using simple (modified integrate-and-fire) neuron models has been achieved elsewhere [33]. Table V lists the neuron parameters that generate the behaviors in Fig. 9. The column "j" is the neuron index, and the other columns correspond to the neuron specification parameters given in Table II and Eqns. (10)-(19). Finally, we note that our approach is not limited to the 20 Izhikevich behaviors in the realm of biologically relevant behaviors. For example, we have generated short-term synaptic facilitation and depression dynamics using the $\gamma_j = 2$ neuron type as a slow adaptive variable to modulate the synaptic efficacy.

V. Discussion

We have presented a dual deterministic/stochastic neural model and have demonstrated that it can reproduce a rich set of synthetic and biologically relevant behaviors. We emphasize the stochastic nature of the neural model, which is used in three places within the neuron: 1. stochastic synaptic integration, 2. stochastic leak, and 3. stochastic thresholds. Algorithmically, stochasticity has a wide variety of uses. Some neural functions compute using stochastic data. Examples include the stochastic

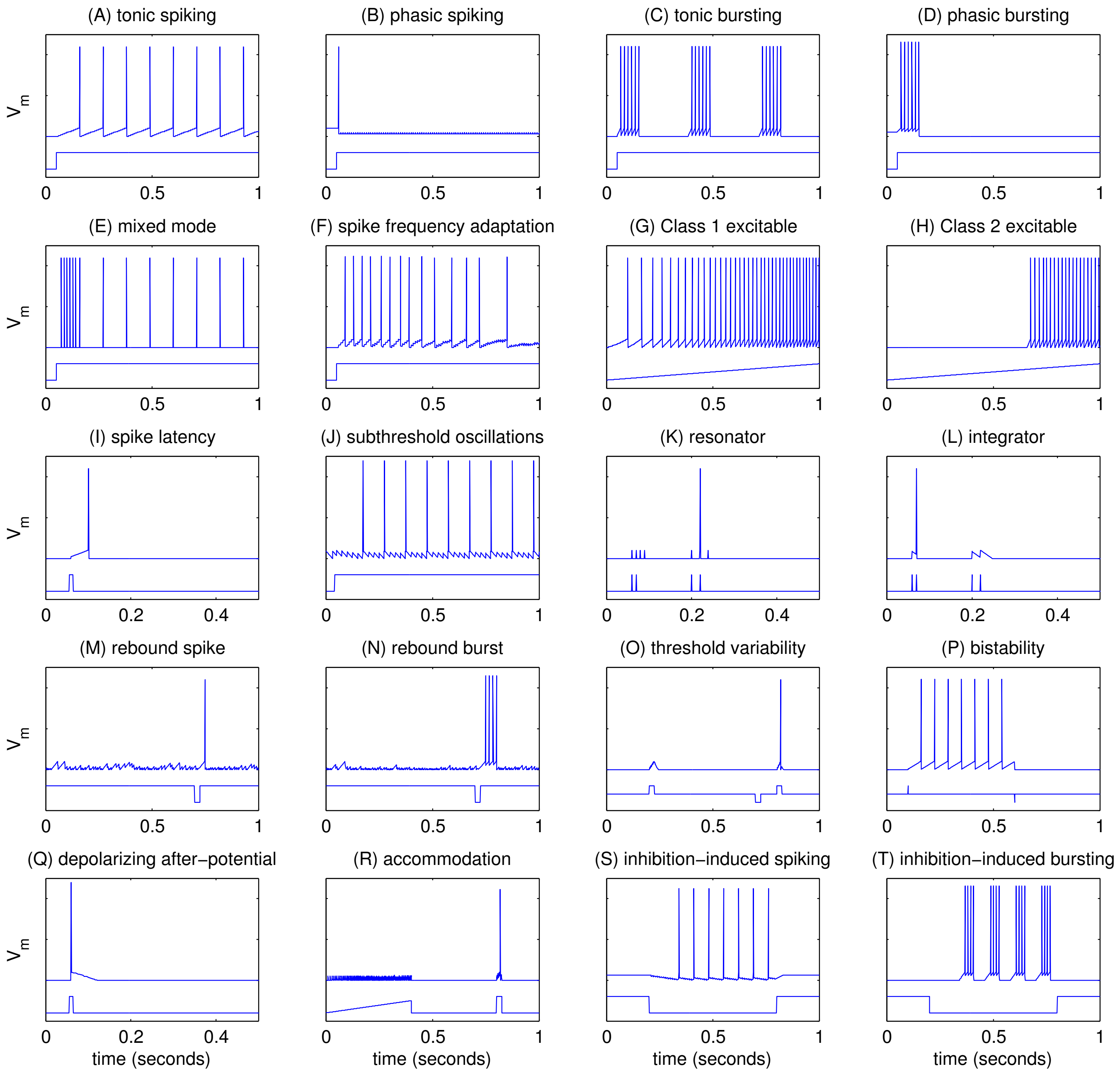

Fig. 9. Twenty biologically relevant behaviors (emulating [24]), generated using our neuron specification.

spiking rate store neuron and its derivatives, computing with probabilities using the noisy-OR and noisy-AND gates [34], as well as sampling algorithms such as Monte-Carlo simulations and Gibbs Sampling. Other algorithms incorporate stochastic values into the computation, such as: Boltzmann machines, noisy gradient descent, simulated annealing, and stochastic resonance. Stochastic values can be used to soften or round-out the non-linear transfer function for some neuron types. Stochastic values also enable fractional leak and synaptic weights values (on average), extending the range of values below a single bit. In stochastic mode, the synaptic weight is the probability, ranging from 2^{-N} to 1 in increments of 2^{-N}, of integrating $+1$ (or -1) on the synapse. Using a random number generator with a uniform distribution, for a large number of samples, the *expected value* of the stochastic synaptic input is a fraction equal to the probability programmed into the synapse.

We have presented a neural model with a wide range of dynamics, and how combining neurons further increases the computational capability. Although the TrueNorth architecture performs a very specific set of computations defined by the neuron model, it is a general purpose architecture in the sense that it can be programmed to perform an arbitrary and wide range of functions. Indeed, it is straightforward to create a universal Turing machine using our neuron model. Using the Boolean logic, memory, and control functions presented in Table IV, we can create the finite automaton and memory required for such a system. However, our goal is not simply

TABLE V.

Name	j	$s_j^{G_i}$	ϵ_j	λ_j	c_j	α_j	β_j	M_j	R_j	κ_j	γ_j
(A) tonic spiking	0	[3,0,0,0]	0	0	0	32	0	0	+0	1	0
(B) phasic spiking	0	[4,20,0,0]	1	+2	0	2	10	0	-15	1	0
(C) tonic bursting	0	[1,-100,0,0]	1	+1	0	18	0	0	+1	1	0
	1	[1,0,0,0]	1	0	0	6	0	0	+0	1	0
(D) phasic bursting	0	[1,-20,0,0]	1	+1	0	18	20	0	+1	1	0
	1	[1,0,0,0]	1	0	0	6	0	0	+0	1	0
(E) mixed mode	0	[3,0,0,0]	1	0	0	32	0	0	+0	1	0
	1	[1,-20,0,0]	1	+1	0	16	20	0	+1	1	0
	2	[1,0,0,0]	1	0	0	6	0	0	+0	1	0
(F) spike frequency adaptation	0	[9,-1,0,0]	1	0	0	32	0	0	+0	1	1
	1	[11,0,0,0]	1	-160	1	1	0	9	+0	1	2
(G) Class 1 excitable	0	[2,0,0,0]	1	0	0	28	0	0	+0	1	0
(H) Class 2 excitable	0	[24,0,0,0]	1	-23	0	3	0	0	+0	1	0
	1	[1,0,0,0]	1	+16	0	256	0	0	+0	1	0
(I) spike latency	0	[10,0,0,0]	1	+1	0	52	0	0	+0	1	0
(J) subthreshold oscillations	0	[22,0,0,0]	0	-1	0	16	30	0	+1	0	0
(K) resonator	0	[2,0,0,0]	0	-1	0	2	0	0	+0	1	0
(L) integrator	0	[24,0,0,0]	0	-1	0	32	0	0	+0	1	0
(M) rebound spike	0	[-1,-16,0,0]	0	+1	0	32	0	0	+0	1	0
	1	[-6,-2,55,0]	0	+64	1	1	100	8	+0	1	2
(N) rebound burst	0	[-1,-16,0,0]	0	+1	0	32	0	0	+15	1	0
	1	[-16,-2,12,0]	0	+128	1	1	100	8	+0	1	2
(O) threshold variability	0	[8,-1,-1,0]	1	-1	0	32	0	0	+0	1	1
	1	[16,-16,-2,0]	0	+64	1	1	100	8	+0	1	2
(P) bistability	0	[1,-100,0,0]	1	+1	0	64	0	0	+1	1	0
(Q) depolarizing after-potential	0	[50,0,0,0]	1	-150	1	35	0	0	+31	1	0
(R) accommodation	0	[22,-2,0,0]	1	-7	0	32	0	0	+0	1	1
	1	[11,0,0,0]	1	-1	0	1	0	9	+0	1	2
(S) inhibition-induced spiking	0	[-13,0,0,0]	1	-1	0	25	50	0	-31	0	0
(T) inhibition-induced bursting	0	[1,-100,0,0]	1	+1	0	28	0	0	+15	1	0
	1	[1,0,0,0]	1	0	0	4	0	0	+0	1	0
	2	[-13,0,0,0]	1	-1	0	12	50	0	-15	0	0

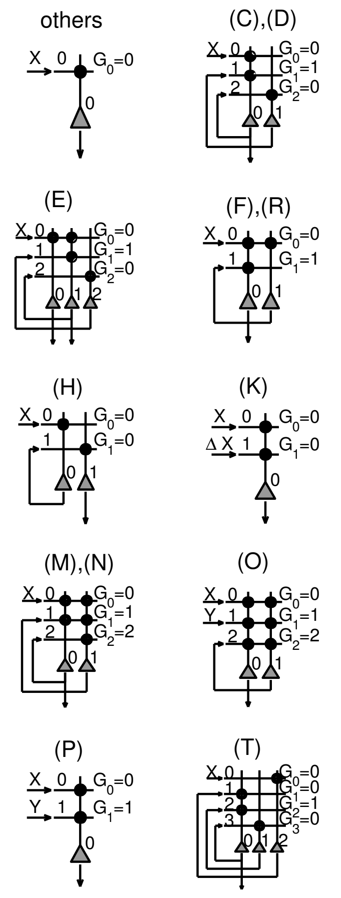

Fig. 10. Parameters for behaviors in Fig. 9: j is the neuron index, and the other parameters are defined in Table II. The diagrams on the right depict the crossbar connectivity, including neurons (triangles), synapses $w_{i,j}$, and axon types G_i. The labels correspond to the alphabetical index of the behavior in the "Name" field in Table V.

simulating a theoretical system, but efficient computation of cognitive algorithms. To that end, applications that exploit the dynamics of the neuron equation, and that utilize the dense connectivity of the synaptic crossbar, run most efficiently on the TrueNorth architecture, as demonstrated in [10].

VI. Conclusion

In this paper, we have presented a digital, reconfigurable, versatile spiking neuron model that supports one-to-one equivalence between hardware and simulation and is implementable using only 1272 ASIC gates (924 gates for the model computation and 348 gates for the random number generator). We demonstrated that the parametric neuron model supports a wide variety of computational functions and neural codes. Further, by combining elementary neuron blocks, we demonstrated that it is possible to synthesize a rich diversity of computations and behaviors. By hierarchically composing neurons or blocks of neurons into larger networks [9], we can begin to construct a large class of cognitive algorithms and applications [10]. Looking to the future, by further composing cognitive algorithms and applications, we plan to build versatile, robust, general-purpose cognitive systems that can interact with multi-modal, sub-symbolic, sensors–actuators in real time while being portable and scalable. In an instrumented planet inundated with real-time sensor data, our aspiration is to build cognitive systems that are based on learning instead of programming, are in everything and everywhere, are essential to the world, and that create enduring value for science, technology, government, business, and society. Advancing towards this vision, we have built chip prototypes, architectural simulators, neuron models and libraries, and a programming paradigm. To realize the TrueNorth architecture in state-of-the-art silicon is the next step.

ACKNOWLEDGMENTS

This research was sponsored by DARPA under contract No. HR0011-09-C-0002. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of DARPA or the U.S. Government. We would like to thank David Peyton for his expert assistance revising this manuscript.

REFERENCES

[1] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, and R. Singh, "Cognitive computing," *Communications of the ACM*, vol. 54, no. 8, pp. 62–71, 2011.

[2] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," in *IEEE Custom Integrated Circuits Conference (CICC)*, Sept. 2011, pp. 1–4.

[3] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, "Compass: A scalable simulator for an architecture for cognitive computing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC 2012)*, Nov. 2012, p. 54.

[4] J. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman, "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *IEEE Custom Integrated Circuits Conference (CICC)*, Sept. 2011, pp. 1–4.

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

[6] T. M. Wong, R. Preissl, P. Datta, M. Flickner, R. Singh, S. K. Esser, E. McQuinn, R. Appuswamy, W. P. Risk, H. D. Simon, and D. S. Modha, "10^{14}," IBM Research Divsion, Research Report RJ10502, 2012.

[7] D. S. Modha and R. Singh, "Network architecture of the long distance pathways in the macaque brain," *Proceedings of the National Academy of the Sciences USA*, vol. 107, no. 30, pp. 13 485–13 490, 2010.

[8] E. McQuinn, P. Datta, M. D. Flickner, W. P. Risk, D. S. Modha, T. M. Wong, R. Singh, S. K. Esser, and R. Appuswamy, "2012 international science & engineering visualization challenge," *Science*, vol. 339, no. 6119, pp. 512–513, February 2013.

[9] A. Amir, P. Datta, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, W. Risk, M. Flickner, R. Alvarez-Icaza, E. McQuinn, B. Shaw, N. Pass, and D. S. Modha, "Cognitve computing programming paradigm: A corelet language for composing networks of neuro-synaptic cores," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013.

[10] S. K. Esser, A. Andreopoulos, R. Appuswamy, P. Datta, D. Barch, A. Amir, J. Arthur, A. S. Cassidy, P. Merolla, S. Chandra, N. Basilico, S. Carpin, T. Zimmerman, F. Zee, M. Flickner, R. Alvarez-Icaza, J. A. Kusnitz, T. M. Wong, W. P. Risk, E. McQuinn, and D. S. Modha, "Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013.

[11] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 115–133, 1943.

[12] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[13] L. Abbott *et al.*, "Lapicques introduction of the integrate-and-fire model neuron (1907)," *Brain Research Bulletin*, vol. 50, no. 5, pp. 303–304, 1999.

[14] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[15] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysical Journal*, vol. 1, no. 6, pp. 445–466, 1961.

[16] R. Stein, "A theoretical analysis of neuronal variability," *Biophysical Journal*, vol. 5, no. 2, pp. 173–194, 1965.

[17] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophysical Journal*, vol. 35, no. 1, pp. 193–213, 1981.

[18] G. Ermentrout and N. Kopell, "Parabolic bursting in an excitable system coupled with a slow oscillation," *SIAM Journal on Applied Mathematics*, vol. 46, no. 2, pp. 233–253, 1986.

[19] R. Rose and J. Hindmarsh, "The assembly of ionic currents in a thalamic neuron i. the three-dimensional model," *Proceedings of the Royal Society of London. B. Biological Sciences*, vol. 237, no. 1288, pp. 267–288, 1989.

[20] C. F. Stevens and A. M. Zador, "Novel integrate-and-fire-like model of repetitive firing in cortical neurons," *Proceedings of the 5th Joint Symposium on Neural Computation*, 1998.

[21] H. Wilson, "Simplified dynamics of human and mammalian neocortical neurons," *Journal of Theoretical Biology*, vol. 200, no. 4, pp. 375–388, 1999.

[22] G. Smith, C. Cox, S. Sherman, and J. Rinzel, "Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model," *Journal of Neurophysiology*, vol. 83, no. 1, pp. 588–610, 2000.

[23] E. M. Izhikevich, "Resonate-and-fire neurons," *Neural Networks*, vol. 14, no. 6-7, pp. 883–894, 2001.

[24] E. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[25] N. Fourcaud-Trocmé, D. Hansel, C. Van Vreeswijk, and N. Brunel, "How spike generation mechanisms determine the neuronal response to fluctuating inputs," *The Journal of Neuroscience*, vol. 23, no. 37, pp. 11 628–11 640, 2003.

[26] R. Jolivet, T. Lewis, and W. Gerstner, "Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy," *Journal of Neurophysiology*, vol. 92, no. 2, pp. 959–976, 2004.

[27] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of Neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.

[28] S. Mihalas and E. Niebur, "A generalized linear integrate-and-fire neural model produces diverse spiking behaviors," *Neural Computation*, vol. 21, no. 3, pp. 704–718, 2009.

[29] N. Imam, F. Akopyan, J. Arthur, P. Merolla, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using event-driven qdi circuits," in *IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 2012, pp. 25–32.

[30] D. Durstewitz, J. Seamans, and T. Sejnowski, "Neurocomputational models of working memory," *Nature Neuroscience*, vol. 3, pp. 1184–1191, 2000.

[31] U. Rutishauser and R. Douglas, "State-dependent computation using coupled recurrent networks," *Neural Computation*, vol. 21, no. 2, pp. 478–509, 2009.

[32] A. Steimer, W. Maass, and R. Douglas, "Belief propagation in networks of spiking neurons," *Neural Computation*, vol. 21, no. 9, pp. 2502–2523, 2009.

[33] W. Gerstner and R. Naud, "How good are neuron models?" *Science*, vol. 326, no. 5951, pp. 379–380, 2009.

[34] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.