Towards Scalable Digital Modeling of Networks of Biorealistic Silicon Neurons

Swagat Bhattacharyya†, *Graduate Student Member, IEEE,* Praveen Raj Ayyappan†, *Graduate Student Member, IEEE,* and Jennifer O. Hasler*, *Senior Member, IEEE*

***Abstract*—The study of biorealistic neuron circuits has been limited by the efficiency of digital implementations. Efficient digital approaches generally use I&F variants, losing important neural properties for network computation. In contrast, accurate neuron ODEs tend to utilize computationally intensive operations, causing the overhead to become prohibitive for large spiking neural network applications. This effort presents efficient digital approximations for coupled HH neurons derived from transistor-channel neural modeling. Neuron models are implemented in C using floating-point and 32-bit fixed-point arithmetic, and small networks are simulated using a fixed-step ODE solver. Our approach enables large network simulation of HH-like neurons, facilitating scalable digital modeling while also providing a direct path towards a framework for analog computation.**

***Index Terms*—HH Neuron, Analytic Model, Fixed-Point, Spiking Neural Net, Synfire, Winner-Take-All, 100K Neuron.**

I. Issues Plaguing Accurate Neuron Models

THE exploration and application of biorealistic neurons have been hindered by the steep tradeoff between computational simplicity and biorealistic dynamical behavior, resulting in delayed or unexplored innovations in real-time neural computation and neuroscience exploration. Existing efficient neuron implementations on digital hardware often utilize I&F variants (all acronyms are introduced in Table I), which often overlook important dynamical properties in network computation applications, leading to reduced efficacy.

The HH model remains a gold standard for biorealistic neuron modeling, especially in applications like neural path planning applications [1], [2]. HH neuron spike shapes can be important for studying network desynchronization, and such biorealistic models are important for determining relationships between connectivity and behavior [3]. Large-scale networks of HH neurons can facilitate the study of brain dynamics; by using 121,000 HH neurons, [4] reproduced experimentally observed behavior in $0.3\,\mathrm{mm}^3$ of neocortical tissue. The HH neuron also offers valuable insights into dendritic integration and synaptic plasticity [5]. However, computationally intensive operations in the HH model, such as natural exponents and division, make it impractical for resource-constrained applications or large-scale emulation on digital platforms.

This work was partially supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2039655. The authors thank Pranav Mathews for useful insight and Jaron Rosenberg for developing a Python implementation. †The first two authors contributed equally to this work. *Asterisk indicates corresponding author*

All authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (correspondence: jennifer.hasler@ece.gatech.edu).

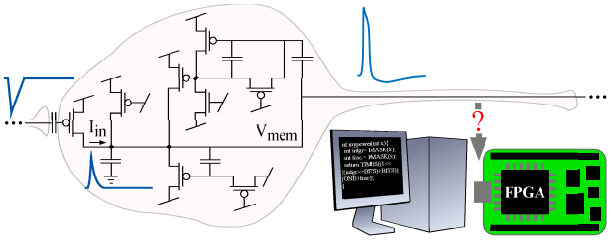

Fig. 1. Prior work [6], [7] suggests biological neurons can be parsimoniously modelled with continuous-time analog circuits. This work aims to bridge the gap between these analog circuits and their digital representation, providing a scalable framework for the emulation of biorealistic silicon neural networks.

To address these challenges, [6] introduced an analog silicon neuron model (hereafter referred to as the FH model), mirroring the voltage clamp responses in the original HH experiments. By leveraging the shared Boltzmann statistics governing carrier transport in MOSFET and biological channels, the six-transistor FH model achieves a more energy-efficient and compact circuit representation than approaches which attempt to directly implement the original HH equations. Despite arriving at entirely different equations than the original HH model, the FH model replicates the bifurcation structure of the HH model given suitable parameters [8]. Yet, the scalability of analog networks of HH-like neurons remains uncertain, and reconfigurable analog chips for large network exploration are not yet commercially available.

Thus, a digital emulation framework for analog systems is sought to guide design until analog platforms become accessible. Previous approaches have attempted to economize computation through FXP arithmetic and parallel FPGA implementation. For example, [9] can potentially simulate 15,000 HH neurons using their FXP approach. Emulation approaches like FPGAs also enable the evaluation of system efficacy in real-time neuromorphic applications, with the understanding that trained networks can be ported to analog platforms when available, provided the digital modeling is accurate. Control applications can be particularly sensitive to delay, and one such application which would benefit greatly from FPGA-accelerated networks is closed-loop neuroprosthesis [10].

This work addresses the need for efficient digital emulation frameworks for silicon neurons by presenting efficient FXP and FP digital approximations for coupled FH neurons [Fig. 1]. Our holistic approach derives mathematical neuron models from circuit-level principles, elucidates operation in networks, and proposes strategies for efficient computation. Our work presents several novel contributions compared to previous work with the FH neuron [6], [8], such as:

- Capability to emulate moderately sized networks of FH

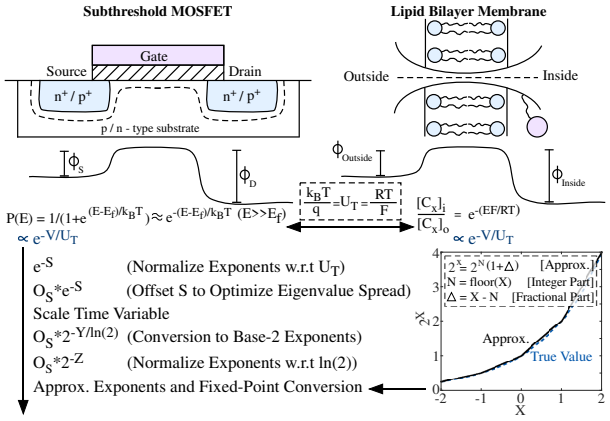

Fig. 2. There are striking similarities between the carrier energy band diagram of transistor and biological channels, which lead to the emergence of Boltzmann factors in both cases. In particular, Boltzmann factors appear in the channel current expressions for MOSFET channels, leading to computational expense. To improve computational expense, we perform a sequence of operations, starting with normalizing by the thermal voltage (U_T) and ending with piecewise approximation of exponents in a base-2, fixed-point representation.

neurons (1D and 2D synfire chains and WTA networks) with both excitatory and inhibitory connections.

- An FXP model of the FH neuron with computation accuracy and efficiency analysis; our approach achieves a better tradeoff between computation overhead and error compared to previous FXP approaches for HH neurons.
- Improved FH neuron modeling through inclusion of the Ohmic region of the channel MOSFETs and inclusion of additional parasitic capacitances in the Na channel gating circuit (for proper bandpass frequency response).
- Model availability (MATLAB, Python, and C) on GitHub: PraveenRajAyyappan/EfficientAnalogNeuron-SNNsims.

Our approach ensures that models can be both directly translated to sampled embedded computing systems while also remaining accurate to physically realizable, biorealistic silicon neurons. Hence, we pave the path for a systematic investigation into the network behavior of biorealistic analog silicon neurons within large-scale, biorealistic network architectures [11], [12] for real-time, energy-efficient neuromorphic computing and for computational neuroscience acceleration [4].

Starting from transistor-channel models for neurons and synapses, we renormalize nodal voltages as fractions of thermal voltage, converting the exponential terms in subthreshold MOSFET channel current expressions into base-2 exponents. These exponents are then approximated using bit shifts and low-order residue multiplications. The conditioned neuron models are implemented in C using both FP and 32-bit FXP arithmetic. We demonstrate the potential of our approach for larger network emulation on commercially available digital infrastructure through small-scale neural circuit experiments.

II. Implementations of Spiking Neural Networks

SNN implementation success is contingent on several interconnected factors, including the use case, neuron model, hardware framework, and computing resource constraints.

A. Neuron Simulations and Specialized Hardware

SNNs primarily serve two purposes: emulating biological networks and solving complex computational tasks (such as robot control, path-planning, and image processing) [14]. The neuron model employed is contingent on the application.

TABLE I
List of Acronyms in This Work

Acronym	Definition	Acronym	Definition
ASIC	Application-Specific Integrated Circuit	GPU	Graphics Processing Unit
		HH	Hodgkin-Huxley [13]
CORDIC	COordinate Rotation DIgital Computer	LUT	LookUp Table
		MAC	Multiply-ACcumulate
CPU	Central Processing Unit	ML	Machine Learning
DAE	Differential-Algebraic Eq.	MOS	Metal-Oxide-Semiconductor
DC	Direct Current	NLP	NonLinear Programming
DSP	Digital Signal Processor	NMAE	Normalized Mean Absolute Error
EKV	Enz-Krummenacher-Vittoz		
FET	Field-Effect Transistor	NRMSE	Normalized Root-Mean-Square Error
FF	Flip-Flop		
FG	Floating-Gate (transistor)	ODE	Ordinary Differential Equation
FH	Farquhar-Hasler [6]		
FOM	Figure Of Merit	PCHIP	Piecewise Cubic Hermite Interpolating Polynomial
FP	Floating-Point		
FPGA	Field-Programmable Gate Array	RTL	Register-Transfer Level
		SNN	Spiking Neural Network
FXP	FiXed-Point	WTA	Winner-Take-All

The classic HH model [13], which is the most biorealistic yet computationally intensive, is generally used for neuroscience research. Conversely, models used for computational tasks, such as the leaky I&F [15] and Izhikevich models [16], abstract away ion channel dynamics, resulting in lower computational cost; indeed, the I&F model has enabled [17] to model 100,000 granular cells and 1,000 golgi cells on an FPGA for studying timing control. However, such abstractions come at the cost of biorealism, which limit the ability of simplified models to capture nonideal ion channel phenomena like temperature dependence in kinetics or channel abnormalities (the physiological basis of several diseases). Some simplified neural models, like the Morris-Lecar model [18] (which biorealistically describes oscillatory behavior in barnacle muscle fibers), are inadequate for modeling spike frequency adaptation [19]. Thus, the dynamical richness of the HH model makes it desirable for both types of SNN applications, although understanding which specific aspects are vital requires further exploration. Before the HH model can be effectively used for computational SNN exploration, a satisfactory approximation requiring significantly fewer hardware-intensive computations must be developed. Such approximations are crucial to make the HH model more competitive with the widely used leaky I&F model, which has 24-fold fewer computations [14], [20].

Software simulation frameworks, such as NEURON, NEST, PCSIM, MOOSE, and GENESIS, have been pivotal, employing parallelized approaches like MPI, multiprocessing, OpenMP, and CUDA to accelerate simulations. While frameworks like NEURON and NEST, which are grounded in higher-order variable-step-size ODE solvers, are adept at balancing solution accuracy and time for large networks, they do not guarantee real-time (bounded) solutions, limiting use in controls applications. In contrast, reconfigurable digital hardware (e.g., FPGAs) can simulate analogous networks in real time with lower power. Reconfigurable digital implementations often use fixed-step solvers, which attain sufficient accuracy given bounded, synchronized inputs, though they require more derivative evaluations than variable-step methods. Current limitations of software frameworks arise from their

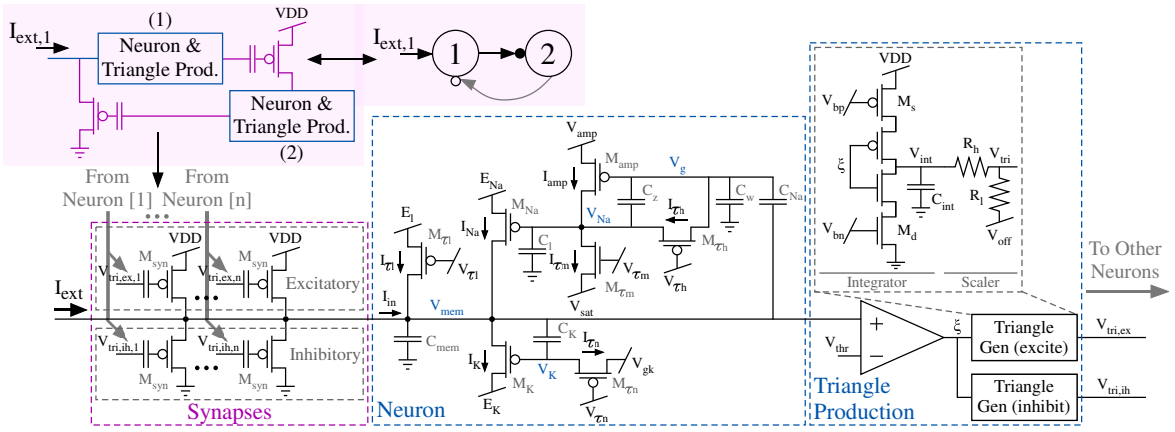

Fig. 3. Circuit-level implementation of networks using the FH model. Neurons are coupled with synaptic circuits, including triangle generators and gate-coupled FG FET synapses. The ideal resistive divider in the triangle generator scaling circuit is merely an abstraction and would be composed of FETs in a complementary MOS implementation. These circuits, which we compactly model, serve as the building blocks for biorealistic analog neural networks [12].

reliance on general-purpose hardware like CPUs or GPUs, whereas barebones SNNs on reconfigurable hardware demonstrate considerably better performance-per-Watt and speed [9]. In short, reconfigurable digital hardware and digital computers serve different purposes, and each presents unique strengths and weaknesses — complicating a direct comparison. Hence, the choice of computing hardware needs careful examination.

B. Selection of Computing Hardware

SNN computing hardware choice is influenced by the application scope, choice of neuron model, and the network size. The implementation overhead of SNNs vary among different neuron models; the HH model, with its four coupled stiff ODEs containing exponential nonlinearities, is particularly intensive. The choice between FP and FXP models is also important. FP models, with their large dynamic range, are well-suited for general computing tasks and are tolerant to ill-scaled variables, but require more memory and processing resources. In contrast, FXP models are more memory-efficient and faster, as they exploit bit shifts and additions to implement low-order approximations of intensive operations (or entirely avoid these operations in some cases); nevertheless, FXP operations can inflate numerical errors, which should be characterized.

GPUs are commonly used for FP models due to their dedicated FP units and ample memory. However, FPGAs offer better optimization in data flow and arithmetic precision for FXP models, as well as native support for peripherals (making them ideal for closed-loop applications like neural prostheses [10]). DSPs can also be used for FXP implementations, offering higher portability, speed, and energy efficiency compared to FPGAs, but their limited resources restrict network scaling.

High-performance GPU clusters are commonly used for parallelization while modeling the brain. For edge computing applications, GPU power draw is generally too high, and ASICs are not commonly pursued due to their low reconfigurability and time-consuming design process. An FPGA provides an appealing middle ground between ASICs and GPUs for smaller computational tasks (i.e., <15,000 HH neurons [9]).

C. Adapting HH Models to an FPGA

The transistor-channel-based FH model is implemented in C and can utilize FXP arithmetic. FPGAs are ideal for our HH-like model given their ability to leverage FXP arithmetic and high-level synthesis tools that translate C to RTL code. Efficient FPGA implementations of the canonical HH neuron model typically use three numerical strategies: CORDIC, piecewise-linear approximations, and orthonormalization.

[21] has optimized the HH model by using LUTs to store approximations of $\tan^{-1}$ and tabulating exponentials of integers within a range. To efficiently compute arbitrary exponents, stored values in the LUT are combined with exponentials of the corresponding fractional digits as computed though CORDIC. In this way, [21] can simulate 120 neurons. In contrast, [9] has formulated the exponentially dependent parameter equations in the HH neuron model using hyperbolic functions realized through CORDIC algorithms in FPGA, simulating 1,034 neurons and projecting that 15,000 neurons can be implemented if FPGA resources are fully utilized.

[22] has used piecewise linear approximations for exponential-dependent functions to reduce computational resource utilization and has simulated 4,096 neurons. The CORDIC algorithm is more accurate than the piecewise-linear approximation method [9], [21]. [23] has proposed an HH model using piecewise-linear approximations to implement divisions and nonlinear internal model variables. [23] also uses base-2 exponents to approximate a hyperbolic formulation of the HH model, avoiding multiplications. [24] has achieved low resource utilization through a reduced-order, two-variable piecewise-linear spiking neuron inspired by HH dynamics.

[25] has shown a multiplier-less HH model leveraging base-2 operations to convert nonlinearities to base-2 exponentials. [25] then used Gram-Schmidt orthonormalization to represent dynamical variables in a base-2 exponential basis, ensuring a scalable, low-power implementation with no DSP blocks.

Most notably, [26] has compared neuron models using different approximation techniques, finding linear interpolation of base-2 exponents more resource-efficient than CORDIC algorithms. Therefore, our approach uses linear interpolation and optional low-order residue multiplications to implement base-2 exponents, avoiding divisions entirely.

III. Silicon Neuron and Synapse Modeling

In this section, we derive ODEs in state variable formulation for biophysically inspired neuron [6] and synapse [7] circuits.

An overview of these circuits, as well as their interconnections, is provided in Fig. 3; in essence, gate-coupled FG FET synapses intercouple neurons, and triangle generator circuits interface with synapse control gates to mimic the Rall Alpha function and provide biorealistic postsynaptic currents given a presynaptic spike. The rest of this section is structured as follows: Subsection III-A extracts explicit neuron ODEs via current balance DAEs and back-substitutions, Subsection III-B models triangle generator response to a neural spike, and Subsection III-C derives a complete expression for the input current to a neuron. We provide useful approximations where appropriate (using circuit-level intuition) and separate terms into static and dynamic quantities to accelerate computation.

A. Neuron

The DAE system for the FH neuron [6] can be extracted by balancing currents on each node of the neuron circuit: the membrane potential (V_{mem}), the potassium potential (V_K), the sodium potential (V_{Na}), and the gating voltage (V_g) and isolating the time-dependence of each nodal voltage:

$$\dot{V}_{mem} = \frac{C_{Na}\dot{V}_g + C_K\dot{V}_k + I_{in} + I_l + I_{Na} - I_K}{C_{mem} + C_{Na} + C_K}, \tag{1}$$

$$\dot{V}_K = \dot{V}_{mem} - I_{\tau n}/C_K, \tag{2}$$

$$\dot{V}_{Na} = \Big(C_z\dot{V}_g + I_{amp} + I_{\tau h} - I_{\tau m}\Big)/(C_z + C_l)\,, \tag{3}$$

$$\dot{V}_g = \Big(C_{Na}\dot{V}_{mem} + C_z\dot{V}_{Na} - I_{\tau h}\Big)/(C_{Na} + C_z + C_w)\,. \tag{4}$$

Eq. 1-4 provide a concise representation; however, they are implicit in the differentials and thus not suitable for explicit solution on an FPGA. To make Eq. 1-4 more explicit, a series of back-substitutions are performed so that each differential is only a function of the input and FET channel currents. We begin by substituting Eq. 3 into Eq. 4:

$$\dot{V}_g = \Big(C_{Na}(C_z + C_l)\dot{V}_{mem} + C_z(I_{amp} - I_{\tau m}) - C_l I_{\tau h}\Big)/C_\alpha^2, \tag{5}$$

$$\text{where} \quad C_\alpha^2 := (C_{Na} + C_z + C_w)(C_z + C_l) - C_z^2. \tag{6}$$

We then substitute Eq. 2 and Eq. 5 into Eq. 1 and simplify:

$$\dot{V}_{mem} = \left(C_{Na}C_z(I_{amp} - I_{\tau m}) - C_{Na}C_l I_{\tau h})\right)/C_\beta^3 + C_\alpha^2(I_{in} + I_l + I_{Na} - I_K - I_{\tau n})/C_\beta^3, \tag{7}$$

$$\text{where} \quad C_\beta^3 := (C_{mem} + C_{Na})C_\alpha^2 - C_{Na}^2(C_z + C_l). \tag{8}$$

We can sequentially continue the back-substitution and recast equations into a state variable formulation. State space approaches are popular in linear systems analysis and provide insight into the FH neuron as will be further explained in Section IV. Normalizing all nodal and supply voltages by the thermal voltage U_T (so $V_{1,:} = V_:/U_T,\ V_{1,:} \to V_:$), we obtain:

$$\dot{\boldsymbol{V}} = \begin{bmatrix}\dot{V}_{mem} & \dot{V}_K & \dot{V}_g & \dot{V}_{Na}\end{bmatrix}^T = \boldsymbol{\Psi}\boldsymbol{I}/U_T; \text{ where} \tag{9}$$

$$\boldsymbol{I} = \begin{bmatrix}I_{in} & I_l & I_{Na} & I_{amp} & I_{\tau m} & I_K & I_{\tau n} & I_{\tau h}\end{bmatrix}^T, \text{ and} \tag{10}$$

$$\boldsymbol{\Psi} = \begin{bmatrix} c_{11} & c_{11} & c_{11} & c_{14} & -c_{14} & -c_{11} & -c_{11} & -c_{18} \\ c_{11} & c_{11} & c_{11} & c_{14} & -c_{14} & -c_{11} & -c_{27} & -c_{18} \\ c_{31} & c_{31} & c_{31} & c_{34} & -c_{34} & -c_{31} & -c_{31} & c_{38} \\ c_{14} & c_{14} & c_{14} & c_{44} & -c_{44} & -c_{14} & -c_{14} & -c_{48} \end{bmatrix}, \text{ wherein} \tag{11}$$

$$c_{ij}: \begin{cases} c_{11} = C_\alpha^2/C_\beta^3 \\ c_{14} = C_{Na}C_z/C_\beta^3 \\ c_{18} = C_{Na}C_l/C_\beta^3 \\ c_{27} = C_\alpha^2/C_\beta^3 + 1/C_K \\ c_{31} = C_{Na}(C_z + C_l)/C_\beta^3 \\ c_{34} = C_z/C_\alpha^2 + C_{Na}^2C_z(C_z + C_l)/C_\alpha^2C_\beta^3 \\ c_{38} = C_l/C_\alpha^2 + C_{Na}^2C_l(C_z + C_l)/C_\alpha^2C_\beta^3 \\ c_{44} = \big(1 + C_z^2/C_\alpha^2\big)/(C_z + C_l) + C_{Na}^2C_z^2/C_\alpha^2C_\beta^3 \\ c_{48} = \big(C_lC_z - C_\alpha^2\big)/C_\alpha^2(C_z + C_l) + C_{Na}^2C_lC_z/C_\alpha^2C_\beta^3 \end{cases} \tag{12}$$

Since neuron FETs operate subthreshold, $\boldsymbol{I}$ is algebraically computed from EKV models [27], where the channel currents for a pFET and an nFET are (respectively)

$$I_p(V_s, V_g, V_d, W/L) = I_{0,p}\,(W/L)\big[e^{-\kappa_p V_g}\big(e^{V_s} - e^{V_d}\big)\big], \tag{13}$$

$$I_n(V_d, V_g, V_s, W/L) = I_{0,n}\,(W/L)\big[e^{\kappa_n V_g}\big(e^{-V_s} - e^{-V_d}\big)\big], \tag{14}$$

where V_s, V_g, and V_d, denote the voltages on the source, gate, and drain terminals (each normalized by U_T); W/L denotes the FET dimension ratio; and κ_p and κ_n denote the gate-channel coupling coefficient of a pFET and an nFET, respectively. Furthermore, $I_{0,p}$ and $I_{0,n}$ can be expressed as

$$I_{0,p} = I_{th,p}\exp\left(\kappa_p\left(\text{VDD} - V_{t0,p}\right) - \text{VDD}\right), \tag{15}$$

$$I_{0,n} = I_{th,n}\exp\left(-\kappa_n V_{t0,n}\right), \tag{16}$$

where $I_{th,p}$ and $I_{th,n}$ denote the pFET and nFET threshold current, respectively. In Eq. 13 and Eq. 14, positive currents flow from source to drain (pFET) and from drain to source (nFET), respectively. Denoting an element-wise product as $\odot$, for the sign convention in Fig. 3, $\boldsymbol{I} \approx \boldsymbol{\zeta} \odot \tilde{\boldsymbol{I}}$, where

$$\boldsymbol{\zeta} = \begin{bmatrix} 1 \\ M_{\tau l}I_{0,p}e^{-\kappa_p V_{\tau l} + E_l} \\ M_{Na}I_{0,p}e^{E_{Na}} \\ M_{amp}I_{0,p}e^{V_{amp}} \\ M_{\tau m}I_{0,n}e^{\kappa_n V_{\tau m} - V_{sat}} \\ M_K I_{0,p} \\ -M_{\tau n}I_{0,p}e^{-\kappa_p V_{\tau n} + V_{gk}} \\ M_{\tau h}I_{0,p}e^{-\kappa_p V_{\tau h}} \end{bmatrix}, \tilde{\boldsymbol{I}} = \begin{bmatrix} I_{in} \\ 1 - \alpha_1 e^{V_{mem}} \\ e^{-\kappa_p V_{Na}}\big(1 - e^{V_{mem} - E_{Na}}\big) \\ e^{-\kappa_p V_g} \\ 1 - \alpha_2 e^{-V_{Na}} \\ e^{-\kappa_p V_K}\big(e^{V_{mem}} - e^{E_K}\big) \\ 1 - \alpha_3 e^{V_K} \\ e^{V_g} - e^{V_{Na}} \end{bmatrix}, \tag{17}$$

where $\alpha_1 = \exp(-E_l)$, $\alpha_2 = \exp(V_{sat})$, and $\alpha_3 = \exp(-V_{gk})$ and where the approximation leverages that $V_{amp} \gg V_{Na} + 3$. If we simplify the expressions for the Na and K channels to neglect the Ohmic region, the waveforms during continuous spiking remain mostly unchanged, but dynamical transitions near the bifurcation points become more pronounced. In Eq. 17, $\boldsymbol{\zeta}$ and $\alpha_:$ are constant and can be precomputed. Consequently, Eq. 9 can be rewritten as follows, consolidating static terms into state transition matrix $\boldsymbol{T}$ to expedite computation:

$$\dot{\boldsymbol{V}} = \boldsymbol{T}\tilde{\boldsymbol{I}}, \text{ where } \boldsymbol{T} = \boldsymbol{\Psi} \odot \left(\boldsymbol{\zeta}\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}\right)^T/U_T. \tag{18}$$

B. Triangle Generator

Triangle generators, which feed into the control gates of FG FET synapses, mimic the behavior of the Rall Alpha function in response to a neural spike [7]. A triangle generator is an asymmetric integrator, where the top (M_{sc}) and bottom (M_d) transistors in the stack (in subthreshold saturation) act

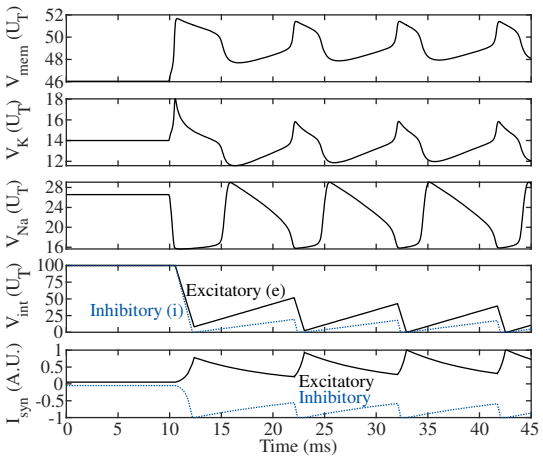

Fig. 4. Neuron nodal voltages, triangle generator nodal voltages, and post-synaptic currents of the FP neuron given a 150 pA step input. At this high spike rate, excitatory post-synaptic currents become stronger via "stacking."

as constant current sources to linearly charge or discharge the integration capacitance (C_{int}) depending on the spike detection comparator state (ξ). When $\xi = 1$ (i.e. during a neuron spike), the stack drains a current of $M_d I_{0,n} e^{\kappa_n V_{bn}}$ out of C_{int}, and when $\xi = 0$, the stack sources a current of $M_{sc} I_{0,p} e^{-\kappa_p V_{bp}+\text{VDD}}$ into C_{int}. Biases V_{bp} and V_{bn} allow for independent tuning of triangle generator attack and decay rates, respectively, aiding exploration into the effect of post-synaptic current rise/fall times on network dynamics.

We simulate two triangle generators for each neuron, where one controls excitatory synapses (via $V_{tri,e}$), and the other controls the inhibitory synapses (via $V_{tri,i}$). While several integration rate configurations perform adequately insofar as the integrated excitatory and inhibitory post-synaptic currents are balanced after synaptic weighting [28], meaningful tunings can help the simple synaptic circuits better emulate the spatial and electrochemical aspects found in biological neural networks. This, when selecting integration rates, it is vital to recognize: (1) in biological neurons, inhibitory connections are more proximal to the soma than excitatory connections [29], and (2) excitatory and inhibitory channels are gated by different molecular species and follow different rate laws [7]. In this work, we set the pull-down and pull-up on the inhibitory triangle generator to be roughly two times faster and eight times slower, respectively, than the excitatory triangle generator as suggested by the figures in [30]. The integrator output voltage in the excitatory and inhibitory triangle generators can be expressed in the following state variable formulation, which can be evaluated entirely using a lookup table: $\left[\dot{V}_{int,e} \dot{V}_{int,i}\right]^T =$

$$\frac{1}{C_{int} U_T} \begin{bmatrix} M_d I_{0,n} e^{\kappa_n V_{bn,E}} & M_s I_{0,p} e^{-\kappa_p V_{bp,E}+VDD} \\ M_d I_{0,n} e^{\kappa_n V_{bn,I}} & M_s I_{0,p} e^{-\kappa_p V_{bp,I}+VDD} \end{bmatrix} \begin{bmatrix} -\xi \\ 1-\xi \end{bmatrix} \quad (19)$$

where output voltages are normalized by U_T. To ensure outputs do not exceed the rails, conditional statements are used in the code. Integrator outputs are then offset and linearly scaled to produce $V_{tri,e}$ and $V_{tri,i}$ before being input into the control gates of FG pFET synapses. In practice, the scaler circuit, which is represented as an ideal resistive voltage divider in Fig. 3, would be composed of FETs in a complementary MOS implementation. These scaler circuits adjust the rail-rail integrator outputs such that $V_{tri,e}$ is close to VDD and $V_{tri,i}$ is close to ground and such that the effective peak-peak voltage on the gates of the synapse FETs is limited between $3U_T$ and $4U_T$. The final triangle generators outputs are given by

$$\begin{bmatrix} V_{tri,e} \\ V_{tri,i} \end{bmatrix} = \frac{R_l}{R_h + R_l} \begin{bmatrix} V_{int,e} \\ V_{int,i} \end{bmatrix} + \begin{bmatrix} V_{off,e} \\ V_{off,i} \end{bmatrix}. \quad (20)$$

C. Synapse

The synapses in our model are represented by control gate-coupled FG pFETs [7]. The sources and drains of excitatory synapses are connected to VDD and the membrane of the post-synaptic neuron, respectively. In contrast, the sources and drains of inhibitory synapses are connected to the membrane of the post-synaptic neuron and ground, respectively. FG pFETs are modeled like normal pFETs, except that the gate voltage is related to the applied control gate voltage (V_{CG}) by: $V_{FG} \approx V_{FG,0} + k_{CG} V_{CG}$, where $V_{FG,0}$ and k_{CG} denote the programmed FG voltage and the capacitive coupling coefficient of the control gate (which generally ranges between 0.7 and 0.8), respectively. Defining an effective coupling factor: $\kappa_f = \kappa_p k_{CG}$, the input current of the n neurons in a network in the presence of an external input current vector ($\boldsymbol{I_{ext}}$) and triangle generator outputs from other neurons is given by

$$\begin{bmatrix} I_{in,1} \\ \vdots \\ I_{in,n} \end{bmatrix} = M_{syn} I_{0,p} (\boldsymbol{W} \odot \boldsymbol{S}) \begin{bmatrix} e^{-\kappa_f V_{tri,ex,1}} \\ e^{-\kappa_f V_{tri,ih,1}} \\ \vdots \\ e^{-\kappa_f V_{tri,ex,n}} \\ e^{-\kappa_f V_{tri,ih,n}} \end{bmatrix} + \boldsymbol{I_{ext}}, \quad (21)$$

where $\boldsymbol{W}$ is a static matrix whose elements represent both network connectivity and "synaptic strength:"

$$\boldsymbol{W} = \begin{bmatrix} e^{-k_p w_{1,1e}} & e^{-k_p w_{1,1i}} & \dots & e^{-k_p w_{1,ni}} \\ e^{-k_p w_{2,1e}} & e^{-k_p w_{2,1i}} & \dots & e^{-k_p w_{2,ni}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-k_p w_{n,1e}} & e^{-k_p w_{n,1i}} & \dots & e^{-k_p w_{n,ni}} \end{bmatrix}, \quad (22)$$

where the values of $w_{i,j:}$ correspond to the programmed FG voltage normalized by U_T with lower values indicating stronger synaptic connections. Connections with large $w_{i,j}$ values ($\geq$ VDD), or no connection at all, are represented by zero elements in $\boldsymbol{W}$. Since neurons are usually either excitatory or inhibitory in function, $\boldsymbol{W}$ is sparse (with less than half of its elements attaining nonzero values). In Eq. 21, $\boldsymbol{S}$ is a scaling matrix that accounts for synapse source dependence. Although both types operate in subthreshold saturation, excitatory synapses have a constant potential (VDD) on their sources while inhibitory synapses have time-varying membrane potentials on their sources. In particular, the channel current of inhibitory synapses scale with $1 - \exp(V_{mem})$; thus, since $\forall i,\ V_{mem,i} \gg 3$, $\boldsymbol{S}$ is approximated by

$$\boldsymbol{S} \approx \begin{bmatrix} e^{\text{VDD}} & -e^{V_{mem,1}} & e^{\text{VDD}} & \dots & -e^{V_{mem,1}} \\ e^{\text{VDD}} & -e^{V_{mem,2}} & e^{\text{VDD}} & \dots & -e^{V_{mem,2}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e^{\text{VDD}} & -e^{V_{mem,n}} & e^{\text{VDD}} & \dots & -e^{V_{mem,n}} \end{bmatrix}. \quad (23)$$

Typical parameters for the neuron and synaptic circuits analyzed in this section (chosen to approximate the time constants

TABLE II
TYPICAL NEURON AND SYNAPSE CIRCUIT PARAMETERS

Transistor Dim. Ratios (μm)				**Voltages** (U_T)								**Capacitances** (fF)				**Other Physical Parameters**			
M_s	2/0.6	M_{Na}	1/0.6	$V_{t0,p}$	28	$V_{bp,i}$	89	V_{sat}	15.4	VDD	100	C_{int}	20	C_l	8	U_T	25 mV	$I_{th,p}$	125 nA
M_d	1/0.6	M_K	1/0.6	$V_{t0,n}$	20	$V_{bn,i}$	7.2	$V_{\tau n}$	-27.2	V_{thr}	47	C_{mem}	800	C_w	8	k_p	0.75	$I_{th,n}$	300 nA
M_{syn}	1/0.6	$M_{\tau n}$	1/0.6	$V_{\tau l}$	18	E_{Na}	52	V_{gk}	14	$V_{\tau h}$	-10.8	C_z	120			k_n	0.67		
$M_{\tau m}$	12/0.6	$M_{\tau h}$	1/0.6	E_l	46	V_{amp}	57.6	E_K	46	$V_{\tau m}$	28.5	C_{Na}	960			R_l	0.5 GΩ		
M_{amp}	12/1.0	$M_{\tau l}$	1/0.6	$V_{bp,e}$	87.2	$V_{bn,e}$	5.2	$V_{off,i}$	10	$V_{off,e}$	90	C_K	400			R_h	9.5 GΩ		

and dynamics in [6], [7]) are presented in Table II. Fig. 4 demonstrates exemplary voltages on the neuron and triangle generator nodes as well as the corresponding post-synaptic currents during continuous spiking.

IV. FIXED-POINT REFORMULATION

The conversion of the FH model from an FP to an FXP representation, as depicted in Fig. 2, involves multiple steps including several changes of variables to make the problem better-posed and function approximations. However, this process introduces numerical errors which must be assessed in relation to the computational overhead.

A. System Scaling and Conditioning

Our formulation is ill-scaled due to the disparity between the small magnitudes of several elements in $\boldsymbol{T}$ and the large exponents in $\boldsymbol{I}$, which in turn results from the resting offset of each nodal voltage. To mitigate this scaling issue, we apply a three-step process. The first step entails performing the following change of variables to the neuron nodal voltages: $V_{2,:} = V_{1,:} - \gamma_{1,:},\ V_{2,:} \to V_{:}$, where $\gamma_{1,:}$ denotes the resting offset (i.e., the DC operating point) of each nodal voltage. Then, the $\gamma_{1,:}$ terms can be factored from the exponentials in $\boldsymbol{I}$, reformulating each exponential as a scaling factor multiplied with an exponential that is unity-valued at DC conditions. Subsequently, we identify and isolate the largest scaling factor within the terms for each row of $\boldsymbol{I}$. We can then implement the third step, which is to absorb the isolated factors into the corresponding columns of $\boldsymbol{T}$. Any remaining scaling factors can then be absorbed into $\alpha_{:}$ constants (if available) or reintegrated into their corresponding exponentials.

Following scaling, $\boldsymbol{T}$ may obtain a large condition number due to differences in capacitances and in FET DC currents. A large condition number, representing a large eigenvalue spread, can potentially introduce substantial numerical errors during derivative computation. We conduct an additional scaling of $\boldsymbol{T}$ to improve its condition number. This step is performed by introducing controlled offsets to each nodal voltage using the solution of an NLP. This corresponds to performing another change of variables while following the same scaling strategy as the previous paragraph: $V_{3,:} = V_{2,:} - \gamma_{2,:},\ V_{3,:} \to V_{:}$. The NLP solved to optimize offsets is formulated as:

$$\gamma_{2,:} := \underset{\boldsymbol{x_{off}}}{\arg\min} \left[\text{cond}\left(\boldsymbol{T}(\boldsymbol{x_{off}})\right)\right]. \tag{24}$$

All elements of $\boldsymbol{T}$ are defined by parameters from the physical analog neuron circuit in Fig. 3, representing inverses of the time constants arising from connecting each FET current source to its corresponding nodes. Extreme magnitude variation among the elements of $\boldsymbol{T}$ seldom arise in properly tuned FH neurons, as time constants will be within the same order of magnitude. Nevertheless, for the parameters specified in Table II, the combined scaling and conditioning approach is valuable in that it reduces the condition number (and hence derivative estimation error) 4.5 times, from 5,084 to 1,112.

Finally, considering that the neuron time constants fall in the millisecond range, we scale the time variable by a factor of 1000 via a change of variables: $t_1 = 1000t,\ t_1 \to t$. This both converts the time unit to millisecond and brings element magnitudes in $\boldsymbol{T}$ closer to unity for convenience of representation. In summary, this comprehensive approach effectively resolves scaling and conditioning issues, providing a more numerically stable solution without increasing implementation difficulty.

B. Fast Base-2 Exponentiation

Eq. 18 includes natural exponents, which are costly for embedded systems. Yet, our application only requires a couple bits of precision, so we can utilize faster, approximate exponentiation approaches. Our solution proceeds by replacing natural exponents with base-2 exponents (utilizing the identity: $e^{V/U_T} = 2^{V/U_T \ln(2)}$) and performing an additional change of variables for U_T-normalized voltages ($V_{2,:} = V_{1,:}/\ln(2),\ V_{2,:} \to V_{:}$), allowing us to rewrite Eq. 18 as:

$$\begin{bmatrix} \dot{V}_{mem} \\ \dot{V}_K \\ \dot{V}_g \\ \dot{V}_{Na} \end{bmatrix} = \tilde{\boldsymbol{T}} \begin{bmatrix} I_{in} \\ 1 - \alpha_1 2^{V_{mem}} \\ 2^{-\kappa_p V_{Na}}\left(1 - 2^{V_{mem} - E_{Na}}\right) \\ 2^{-\kappa_p V_g} \\ 1 - \alpha_2 2^{-V_{Na}} \\ 2^{-\kappa_p V_K}\left(2^{V_{mem}} - 2^{E_K}\right) \\ 1 - \alpha_3 2^{V_K} \\ 2^{V_g} - 2^{V_{Na}} \end{bmatrix};\ \tilde{\boldsymbol{T}} = \frac{\boldsymbol{T}/\ln(2)}{1000}, \tag{25}$$

where voltages are in units of $U_T \ln(2)$. Low-order polynomials can be used to approximate 2^x with an accuracy of $\pm 1\%$.

To ensure continuity in neural responses, we can start with a *continuous*, piecewise approximation for 2^x that linearly interpolates between $2^{\lfloor x \rfloor}$ and $2^{\lceil x \rceil}$ (proposed by [31]):

$$\phi_l(x) = 2^N (1 + \Delta)\,; \text{ where } N = \lfloor x \rfloor,\ \Delta = x - N. \tag{26}$$

In a FXP implementation, we can use bit-masking to find the integral digits (N) and the fractional residue (Δ) and use a bit shift operation to compute 2^N. Eq. 26 unfortunately overestimates the value of 2^x by up to 6% [Fig. 5]. Informed by the error function shape in Fig. 5(a), we introduce an error reduction method that multiplies a correction term that is quadratic in Δ to Eq. 26. Since the error of $\phi_l(x)$ is already small, any correction term should prioritize computational speed over accuracy. Therefore, we can append an empirically derived correction term to Eq. 26 for improved accuracy:

$$\phi_c(x) = 2^N(1 + \Delta)(0.25(\Delta^2 - \Delta) + 1), \tag{27}$$

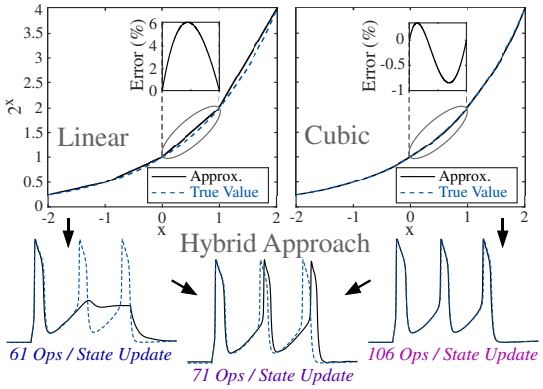

Fig. 5. Comparison of fast linear and cubic approximations of 2^x based on interpolation and empirical error correction. A good trade-off between dynamical accuracy and computational intensity is achieved via a hybrid approach using both exponent approximations.

where multiplication by 0.25 can be performed using a bit-shift operation. Eq. 27 is accurate to $\pm 1\%$, but it is more computationally intensive than Eq. 26 [Fig. 5]. Neuron dynamics can diverge considerably from the FP model, especially near the threshold current, if ϕ_l is exclusively used. In fact, exclusive usage of ϕ_l can cause network dynamics to diverge from the FP model – for example, after exciting a synfire chain, self-sustaining oscillations present in the FP model are no longer observed. In contrast, exclusive usage of ϕ_c yields good dynamical accuracy with a high penalty in the number of operations to update neuron state variables. As shown in Fig. 5, intermixing ϕ_l and ϕ_c yields a "balanced" model that is dynamically accurate with only a slight increase in the number of operations relative to a model using exclusively ϕ_l:

$$\begin{bmatrix} \dot{V}_{mem} \\ \dot{V}_K \\ \dot{V}_g \\ \dot{V}_{Na} \end{bmatrix} = \tilde{\boldsymbol{T}} \begin{bmatrix} I_{in} \\ 1 - \alpha_1 \phi_c(V_{mem}) \\ \phi_l(-\kappa_p V_{Na})(1 - \phi_l(V_{mem} - E_{Na})) \\ \phi_l(-\kappa_p V_g) \\ 1 - \alpha_2 \phi_l(-V_{Na}) \\ \phi_c(V_{mem} - \kappa_p V_K) - \phi_l(E_K - \kappa_p V_K) \\ 1 - \alpha_3 \phi_l(V_K) \\ \phi_l(V_g) - \phi_l(V_{Na}) \end{bmatrix}. \quad (28)$$

Eq. 28 is used for all FXP computations henceforth. Note that ϕ_l suffices for all synaptic computation.

C. Conversion of Floating-Point Models to Fixed Point

To facilitate the creation of FXP models, equivalent bare-bones FP models are created in MATLAB and C. We begin by simulating the approximate ODEs for the neuron and triangle generator FP models in MATLAB and C using a fixed-step, explicit Euler ODE solver, utilizing approximate base-2 exponents to verify functionality. The explicit Euler method (a first-order method) is used due to its simplicity, especially for implementing a sampled, real-time system. Subsequently, the MATLAB Fixed-Point Converter application is employed to determine the range of each variable and to estimate the required number of integer and fractional bits for each constant and variable while maintaining a robust safety margin. Notably, the Fixed-Point Converter is also capable of generating example FXP code for preliminary error analysis.

The use of the Fixed-Point Converter revealed that the explicit Euler solver fails to converge for a 16-bit model given a step input current exceeding the threshold for continuous spiking. At the minimum, the FXP model must preserve the transition in dynamics near the the beginning and end of the input current versus frequency (IF) curve (i.e., starting and stopping behavior) and IF curve monotonicity (discussed further in the subsequent subsection). Throughout this process, the FP model, is used to identify a feasible range of Euler step sizes while finding the minimum bit width for convergence. Once convergence is established, we verify appropriate dynamical behavior for step inputs across the entirety of the IF curve. We found that a 32-bit model convergences with minimal error across the IF curve. Note that even if the computation can be done using fewer number of bits, thereby reducing memory cost, the step size typically needs to shrink substantially to maintain accuracy, significantly increasing computation cost. Given the importance of solution accuracy and speed in this work, we chose to proceed with a 32-bit model representation. Different variables require different representations to prevent overflow. For instance, neuron nodal voltages are represented in Q24.8 format, triangle generator output voltages are in Q16.16 format, and the outputs of the exponential function are represented in Q8.24 format. The optimization of number systems to reduce resource utilization is platform-dependent. Drawing on insights from the Fixed-Point Converter, we developed FXP C models from the barebones FP C models. This approach allows for superior optimization for computation speed over MATLAB-generated C code.

D. Analysis of Model Behavior and Error

Characterizing errors emerging from the conversion from a FP to a FXP representation is of paramount importance. These numerical errors can originate from several sources, including repeated additions (where an optimal step size is critical for the error magnitude [32]), the assigned precision for variable containers, and approximate exponentiation.

The shape of the nodal voltages for the FXP and FP neuron models during continuous spiking are quite similar [Fig. 6(a)]. The responses of both models to step input currents align closely with each other and reflect true biological behavior. For instance, a small step elicits a single spike from the neuron, with a few additional spikes observed upon a slight increase in the step height. Both models have a threshold current, albeit different, beyond which the neuron fires continuously, indicating a stable limit cycle. As the current escalates, there is a concurrent increase in the firing frequency and the minimum voltage during the hyperpolarization phase. However, past a certain input level, the spike rate decreases, and the neuron ceases to spike continuously, as depicted in Fig. 6(b).

The IF curves of the FXP point and FP models, which represent the portion of the input space where a stable limit cycle exists, are shown in Fig. 6(c). The IF curve also helps one tune input synapses to obtain the current needed for the neuron to spike at a given frequency. Notice that the FXP model has a higher starting current and lower ending current, but the overall profile and the monotonicity of the IF curve

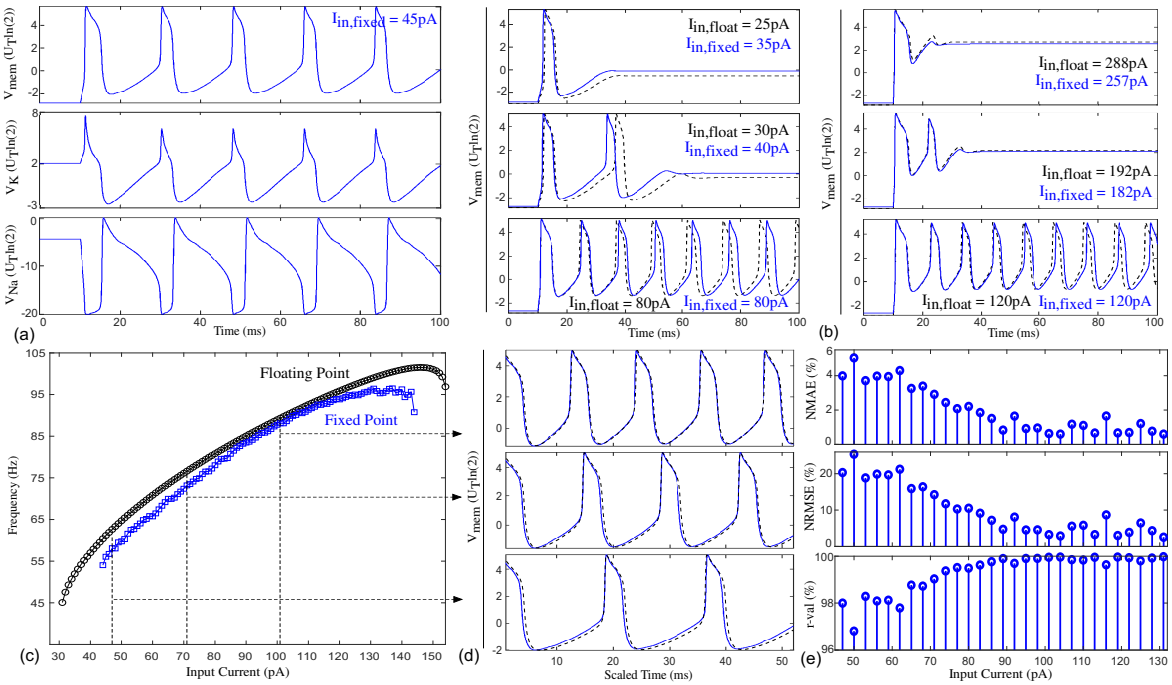

Fig. 6. (a) Nodal voltages of the FXP neuron given a $45\,\mathrm{pA}$ input. (b) Comparisons FP (black, dotted line) and FXP (blue, solid line) model spikes for step inputs of different peak levels. Both models only spike once for small inputs, and the models spike twice when the input peak level is increased slightly. Beyond a threshold level, the neurons spike continuously with a rate which rises with rising input current. Beyond some input current, both models stop firing continuously; the models only fire a few times in this region, mirroring the behavior observed before the start of the continually spiking region. (c) IF curves showing the mean spiking frequency in the continually spiking region with respect to the input current for the FP and FXP models. (d) FP and FXP spikes plotted on scaled time axes for a few input currents such that peaks overlap. Although the spiking frequency differs between the models, overall spike shapes are preserved during conversion from a FP to a FXP representation. (e) Three error metrics (NMAE, NRMSE, and Pearson r-val) for the FXP model as a function of the input current. Error analysis is performed by comparing the FP and FXP model responses on the scaled time axes described in (c).

(its most important characteristics) are preserved between the models. The IF curve indicates that there is a slightly different time scale between the models, but if the time axis of the FP model output is stretched so that the spike peaks of both models overlap [Fig. 6(d)], we find that the shape of the spikes are well-preserved after conversion from FP to FXP.

To systematically characterize errors in the shape of the spikes in scaled time, we introduce three metrics, each selected on its merit to provide unique insights into error characteristics: NMAE, NRMSE, and Pearson correlation coefficient (r-val), which are plotted across the range of the IF curve [Fig. 6(e)]. NMAE characterizes the mean FXP error intuitively, as a percent of the FP output range; NRMSE (RMSE normalized by the standard deviation of the truth) characterizes the FXP error signal energy as a percent of the FP output energy; and r-val characterizes how well the depolarization, repolarization, and hyperpolarization phases align between the FXP and FP model. We have defined NMAE and NRMSE as follows:

$$\text{NMAE} := \frac{\|\boldsymbol{X_{fix}} - \mathcal{F}(\boldsymbol{X_{float}})\|_1 / N}{\max(\boldsymbol{X_{float}}) - \min(\boldsymbol{X_{float}})} \times 100\%, \tag{29}$$

$$\text{NRMSE} := \frac{\|\boldsymbol{X_{fix}} - \mathcal{F}(\boldsymbol{X_{float}})\|_2}{\|\boldsymbol{X_{float}} - \langle \boldsymbol{X_{float}} \rangle\|_2} \times 100\%, \tag{30}$$

where $\boldsymbol{X_{float}}$ and $\boldsymbol{X_{fix}}$ denote the vectors corresponding to V_{mem} at each timestamp for the FP and FXP model, respectively, N denotes the number of points in X_{float}, and $\mathcal{F}$ denotes a function that interpolates the corresponding value of $\boldsymbol{X_{fix}}$ in scaled time. In this work, $\mathcal{F}$ is a PCHIP, since PCHIPs are well-behaved around extrema and edges with minimal overshoot. Overall, we find that shape error is generally low, yet it is the lowest across the center region of the IF curve. Our mean NMAE is lower than the NMAE of other previous approaches [9], [22], [25]. At lower input currents, the slight drop in the correlation between the FP and FXP models signifies an enlarged dissimilarity in action potential phase timing. The resulting distortion in the spike shape can be attributed to the impact of truncation errors on the dynamics of our coupled nonlinear ODE system. Truncation errors are more prominent at smaller input currents where they cause a greater percentage change in the Euler step derivative estimate since derivatives are generally lower at lower spike rates. Nevertheless, our low shape error highlights the effectiveness of our approach in navigating the complex interplay among precision, performance, and biological relevance, despite sources of error during the conversion from a FP to a FXP representation.

V. Networks Utilizing the Fixed-Point Model

In the pursuit of accurate models for biological neural pathways, it is essential to explore network architectures with a focus on biorealism. Adherence to biological principles facilitates efficient resource scaling with neuron count. In contrast to contemporary neuromorphic designs, often characterized by fully connected networks where every neuron can potentially connect to all others [33], which leads to space complexities of $\mathcal{O}\left(n^2\right)$, biological systems present a markedly different architecture. Biological systems typically consist of dynamically rich neurons sparsely and locally connected to approximately 500-5,000 synapses (largely invariant through signal processing hierarchies), resulting in space complexities close to $\mathcal{O}(n)$ [34]. Compute energy cost is closely linked

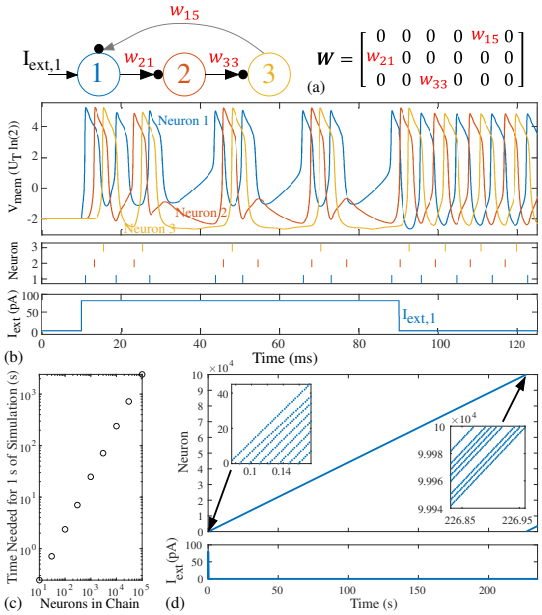

Fig. 7. (a) A recurrent, three-neuron 1D synfire chain with solely excitatory connections. (b) Response of the 1D synfire chain in (a) to an external current pulse. Here, all synapses are homogeneous, and the input current exceeds the maximum synapse channel current. The input causes the neurons to spike in succession; after the input is nulled, the recurrent, positive feedback connection maintains a stable oscillation. (c) Measured execution time to simulate 1 s of 1D synfire chain activity versus network size; the simulations, which confirm the prediction of $\mathcal{O}(n)$ time complexity, use a single core of an Intel i9-13900K CPU at a clock speed of 5.6 GHz. (d) Raster plot of a 1D synfire chain containing 100,000 FH neurons. After excitation, the network cyclically propagates spike fronts; the number of concurrently spiking neurons is an invariant quantity matching the population of initially excited neurons.

to space complexity, and sparse routing schemes contribute significantly to biological energy efficiency [35]. These observations suggest that our proposed approach of using dynamically rich FH neurons and sparse connections would open avenues for energy-efficient large-scale analog neuromorphic processing by mitigating the large energy burden of spike routing in contemporary approaches without digitally assisted addressing [33]. This work initiates exploration of biologically plausible (locally connected) networks of FH neurons by examining the dynamics and scaling of recurrent 1D synfire and feedforward 2D synfire chains with exclusively excitatory synapses, followed by an investigation of a WTA network containing several inhibitory synapses.

A. Recurrent 1D Synfire Chains

Synfire chains, which are prevalent in biological motor pathways, consist of interconnected pools of neurons that enable the sequential transmission of spike packets, forming a chain-like firing pattern. In synfire chains, external input currents excite an input neuron pool, and the outputs of the spike packets generated by the input pool initiate the propagation of spike packets to subsequent pools via predominantly excitatory connections [36]. Synfire chains may also include recurrent connections among layers. Synfire chains are a mode of long-range communication in biological neurons [36] and serve as benchmark for sparse, large-scale neural simulation.

We begin our network exploration with a recurrent 1D synfire chain in which an external input stimulates the first neuron, each neuron subsequently excites the next, and the last neuron reexcites the first, establishing a chain-like sequence

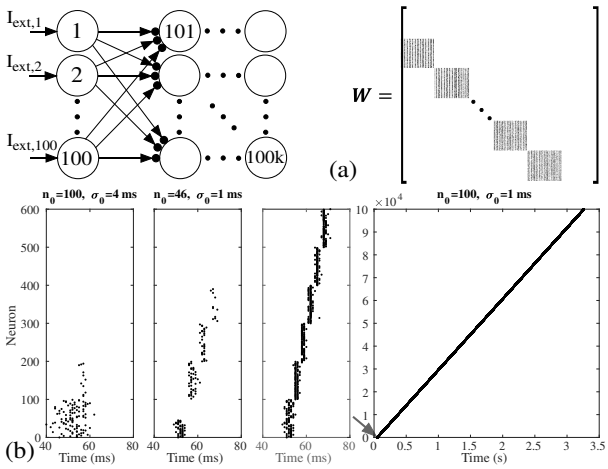

Fig. 8. (a) 100,000-neuron heterogeneous feedforward synfire chain containing 1000 layers with 100 neurons each. Adjacent layers are fully connected with positive normally distributed synaptic weights as depicted in the weight matrix $\boldsymbol{W}$. $\boldsymbol{W}$ is depicted in grayscale, where darker shades indicate stronger connections and pure white indicates an absence of synaptic connections. (b) Raster plot of the response of the network in (a) to a spike packet applied to n_0 neurons of the input layer with arrival times drawn from a normal distribution with a standard deviation of σ_0 and mean of 50 ms. Packets desynchronize either when n_0 is too small for the given σ_0, or when σ_0 is excessive for the given n_0, showing the existence of a bounded region in (σ_0, n_0) space supporting long-term spike packet propagation, as expected from biology.

of firing activity. Recurrent 1D synfire chain containing 3-100,000 FH neurons are demonstrated in Fig. 7 using our FXP neuron model. As verified in Fig. 7(c), simulations of FH neuron 1D synfire chains have a time complexity of $\mathcal{O}(n)$, and the space complexity also scales as $\mathcal{O}(n)$.

B. Feedforward 2D Synfire Chains

Though closely related (with $\mathcal{O}(n)$ space complexity), feedforward 2D synfire chains have higher synaptic density and dynamical richness compared with the recurrent 1D synfire chains. In our demonstration in Fig. 8, we have implemented a heterogeneous 2D synfire chain of 100,000 FH neurons (1000 layers containing 100 neurons) using our FXP model. This network has nearly 10 million synaptic connections (with normally distributed weightage) due to the full connectivity between adjacent layers. Although our network architectures are fixed after initialization in this work, one could also add a noise term to the triangle generator biases and to the synapse FG voltages for modeling stochastic synaptic transmission delays and stochastic synaptic weights, respectively.

In each of our simulations in Fig. 8, we introduce a brief spike packet with normally distributed spike arrival times into a subset of input neurons. Our simulations reveal distinct parametric limits governing the long-term survival of spike packets in heterogeneous synfire chains. We show that spike packet desynchronization occurs under two conditions: either when an insufficient number of neurons receive the input spike packet for the existing level of stochasticity in the spike arrival times, or when the stochasticity in the spike arrival times is excessive for the given number of input neurons receiving the input spike packet. These observations align with established computational neuroscience modeling [37], underscoring the bioaccuracy of the FH neuron and synaptic circuits.

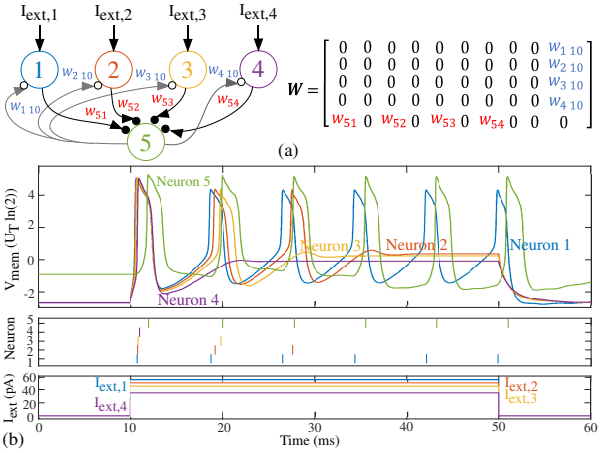

Fig. 9. (a) Five-neuron spiking WTA network. The first four neurons receive external input currents and excite the fifth neuron. The fifth neuron in turn can inhibit all other neurons. (b) Response of the WTA network in (a) to input current pulses with different peak levels using FXP modeling. In a WTA network, the most excited input neurons win by inhibiting the least excited input neuron; inhibition continues until only the input neuron with the highest input current (most excitation) is left spiking. The synapse weights used here are tuned so that each spike of the inhibitory neuron (i.e., neuron 5) causes an "elimination." Note that the current pulses here are set up such that at their peak levels, $I_{ext,1} > I_{ext,2} > I_{ext,3} > I_{ext,4}$.

C. Winner-Take-All Structures

WTA behavior is found in biological networks where one or more neurons remain excited while competing with and inhibiting other neurons in the pool. Winning neurons use nonlinear inhibition to suppress the spiking of other neurons in the pool. WTA behavior can be efficiently emulated in analog circuits and can act as a powerful framework [38] for solving learning and classification problems [39], [40]. WTA was first shown in a non-spiking form by Lazzaro, where the implementation achieved an $\mathcal{O}(n)$ interconnect complexity [41]. We show spiking WTA dynamics with our FXP model by forming a 5-neuron network as shown in Fig. 9(a). In this demonstration, we homogenize the network, so there are only two synaptic weights: excitatory and inhibitory. The excitatory weight is set so that the synapse strength is sufficient to excite the inhibitory neuron through its entire IF curve. We then set the inhibitory weight so the inhibitory neuron eliminates one input neuron with every spike. The simulation results in Fig. 9(b) confirm that the desired WTA function is achieved with $\mathcal{O}(n)$ time complexity. In other words, the number of inhibitory spikes needed to determine the neuron with the largest input current mirrors the number of input neurons. Furthermore, the space complexity also scales as $\mathcal{O}(n)$.

VI. Discussion

This section provides more insight on two topics: (1) the complexity of the FH model in comparison to other neuron models in both FP and FXP representations and (2) the efficacy of our approach of converting from a FP to a FXP representation. We begin the discussion by comparing the FP model of the FH neuron to other previously proposed types of FP neuron models in Table III [20], [42].

The biological plausibility of a model is determined by considering the number of biologically relevant parameters used in the model formulation and how well the model reflects

TABLE III
Comparison of Floating-Point Neuron Models

Neuron Model	State Variables	Bio-realistic	FP Ops Per Step	Max Step Size (ms)	Dig. Power Draw (AU)
Integrate & Fire [15]	1	No	5	1.000	1.00
Izhikevich [16]	2	No	13	1.000	2.60
FHN [43], [44]	2	No	18	0.250	14.4
AdEx [45]	2	No	24	1.000	4.80
Hindmarsh-Rose [46]	3	No	30	0.250	24.0
Wilson [47]	4	No	45	0.250	36.0
Morris-Lecar [18]	2	Yes	60	0.100	120.
HH [13]	4	Yes	120	0.075	320.
FH (Proposed)	**4**	**Yes**	**144**	**0.020**	**1440**

the original HH neuron model, which is widely regarded as the gold standard. Note that the FH model utilizes transistors to represent the ion channel conductance response during the HH voltage clamp experiments and preserves the original time constants (τ_m, τ_h, and τ_n) [6]. Furthermore, the FH model exhibits the same Hopf bifurcation structure as the HH neuron [8], providing additional evidence for its biological plausibility and supporting our proposed formulation.

For large sparse networks, as typical in biology, memory utilization scales with the number of state variables (column 2 of Table III), provided that constants are not replicated. The compute power requirement for implementing a specific model is roughly proportional to the number of FP operations per Euler step and inversely proportional to the maximum (reliable) step size available to a fixed-step-size solver. Table III presents a normalized power metric reflecting this observation.

The digital FH model power metric is inflated by the requirement for its maximum step size. Maximum step size is indicative of the "stiffness" of the corresponding system of ODEs; consistent with other dynamical systems in subthreshold MOS [32], the stiffness of the FH model is inherently high. Compared to the other two biologically plausible models in Table III, the FH model also requires 24 and 84 more FP operations per Euler step than the original HH neuron model [13] and the Morris-Lecar model [18], respectively. Therefore, comparing the FH model to the HH model, it appears that our model is slightly more computationally expensive. However, out of the 144 FP operations, 64 FP operations (32 additions and 32 multiplications) compose a single MAC operation in our model formulation, involving a multiplication of a 4x8 matrix with an 8x1 vector. This approach aligns well with current trends in ML, where ML accelerators incorporate built-in MAC units to make MACs less expensive than expected from a straightforward calculation of the number of FP operations.

The remainder of this section presents a comparative breakdown of the computational power and memory requirements for our FXP model given a generic digital platform to demonstrate that the resource requirements are reasonable compared to previous HH neuron approaches and so that the reader can evaluate maximum network sizes for their platform of choice.

Our FXP model necessitates a global memory of 188 bytes to store computation constants and an additional 68 bytes per neuron to update state variables and other internal variables. Table IV summarizes the computation needed for a parallel one-step state update compared to other implementations.

The HH neuron models compared in Table IV [9], [22], [25] are optimized for FPGAs. Since our model is not

yet FPGA-optimized, we have estimated FPGA-based design computations (LUTs, FFs, DSPs) as CPU-based arithmetic operations for relevant comparisons. [22] has implemented HH neurons with fewer operations through piecewise linear approximations, however, their approach comes with a large increase in data shuffling due to extensive LUT usage. [9] has used CORDIC to efficiently solve parameters and implement divisions yet required many DSP blocks and seven Euler steps per state update. Our approach aligns with that in [25], using FPGA resources (e.g., Add/Sub, Bit shifts, AND, OR, Absolute value, NOT, FFs) for computation. In our model, multiplication units consume one binary multiply and one bit shift, as we implement our model on a CPU, where converting multiplies into logical adds and bit shifts does not provide computational benefits. On FPGAs, binary multiplication can be optimized to logical shifts and add operations [23], converting the 50 multiplies into simple binary arithmetic.

In a network, the maximum step size of our FXP model diminishes relative to the single-neuron FP implementation in Table III. This reduction in step size stems from the increased "stiffness" indroduced by the synaptic circuits, which generate waveforms with sharp transitions and exponential derivatives. It is worth noting that although [25] achieved a much higher step size for neuron ensembles than our approach, much of the improvement can likely be attributed to their implementation, where they synchronously triggered the 150-neuron ensemble with an external input current (without forming a true network with inter-neuron synapses). In contrast, the other approaches compared in Table IV, including ours, incorporate inter-neuron synapses. Within this context, our model demonstrates a network step size comparable to other HH approaches.

Given the variety of performance metrics in Table IV, we define a FOM to equitably assess the efficacy of the conversion from FP to FXP representation via three key considerations:

1) The disparate number of FP operations among base neuron models, reflective of differing bioaccuracy.
2) Inherent tradeoffs between conversion accuracy and computational efficiency in FXP representations, as more accurate models are costlier.
3) The vast differences in compute requirements, such as the necessary clock cycles, among FXP operations.

These considerations drive our FOM to normalize by the number of operations in the corresponding FP model (N_{FPOps}), reward for conversion accuracy (MAE), and penalize for compute overhead (i.e., the total number of clock cycles required per Euler step if operations are performed sequentially – a quantity indicative of the energy cost). Our FOM, which embodies conversion accuracy, compute cost, and the inherent base FP model complexity, is given by

$$\text{FOM} \;:=\; (\text{MAE}/N_{FPOps}) \sum_i \left(N_{Clk/Op,i} N_{Ops,i}\right), \quad (31)$$

where $N_{Clk/Op,i}$ and $N_{Ops,i}$ denote the number of clock cycles for a given type of operation and the number of that type of operation per Euler step, respectively. In our convention, a *lower FOM* corresponds to better conversion efficacy.

In a fixed-point FPGA implementation, "Simple Binary Arithmetic Operations" (i.e., Add, Sub, Bit Shifts, Combinatorial Logic/Gates, Flip Flop) take only 1 clock cycle, whereas "Binary Multiplies" use 5 clock cycles when efficient multiplication methods are used [48]. "Hyperbolic Functions," realized through CORDIC algorithms in FPGAs, take 3 clock cycles [9]. These quantities were used in Eq.31 to calculate the FOM for the proposed FXP FH model and the compared FXP HH models [22], [25] as reported in Table IV. The FOM of our model is 8.3252, which is a significant improvement over the FOM of [25] (11.25) and [22] (9.15). Since [9] does not report MAE, we are unable to calculate the corresponding FOM. These results indicate that our FP to FXP model conversion approach yields a better compromise between computational overhead and error over previous approaches.

TABLE IV
COMPARISON OF COMPUTATIONAL OPERATIONS PER TIMESTEP

Metric	**Proposed**	[25]	[9]	[22]
Simple Bitwise Arithmetic Operations	**252**	267	51	13
Binary Multiplies	**50**	0	42	34
Divisions and Hyperbolic Functions	**0**	0	25	0
Number of Euler Steps per State Update	**1**	1	7	3
Normalized Mean Absolute Error (%)	**2.4**	5	-	6
Mean Pearson Correlation (%)	**99.3**	90.4	96	99
FOM (***Lower is Better***)	**8.33**	11.25	-	9.15
Network Simulation	**Yes**	No	Yes	Yes
Network Step Size (µs)	**1**	7800	31.25	7.8

VII. CONCLUSION

This work presented an analytical approach for modeling networks of biorealistic silicon neurons. By employing state variable formulation, approximate base-2 exponentiation, and proper scaling, our simulations in C using FXP arithmetic compared favorably with FP analogues. Our FP model of the FH neuron, while more expensive than the HH model, converts to FXP more effectively and integrates better with MAC units.

Our model is suitable for FPGAs and other digital platforms, enhancing the feasibility of employing HH-like neurons in computational SNNs and evaluating the impact of biologically accurate neuron dynamics on performance metrics, a process previously hindered by the high cost of the HH model. With the rise of analog system tools and reconfigurable platforms, our models, which accurately describe analog circuits, enable neuromorphic embedded system designers to seamlessly transition from digital emulators to large-scale analog systems.

REFERENCES

[1] S. Koziol, S. Brink, and J. Hasler, "A neuromorphic approach to path planning using a reconfigurable neuron array IC," *IEEE T VLSI Syst*, vol. 22, no. 12, pp. 2724–2737, 2014.

[2] M. Davies et al., "Advancing neuromorphic computing with Loihi: A survey of results and outlook," *P IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[3] L. Timón, P. Ekelmans, N. Kraynyukova, T. Rose, L. Busse, and T. Tchumatchenko, "How to incorporate biological insights into network models and why it matters," *J Physio*, vol. 601, no. 15, pp. 3037–3053, Sep. 2022.

[4] H. Markram et al., "Reconstruction and simulation of neocortical microcircuitry," *Cell*, vol. 163, no. 2, pp. 456–492, 2015.

[5] K. Petousakis, A. Apostolopoulou, and P. Poirazi, "The impact of Hodgkin–Huxley models on dendritic research," *J Physio*, vol. 601, no. 15, pp. 3091–3102, Oct. 2022.

[6] E. Farquhar and P. Hasler, "A bio-physically inspired silicon neuron," *IEEE T Cir Syst I*, vol. 52, no. 3, pp. 477–488, 2005.

[7] C. Gordon, E. Farquhar, and P. Hasler, "A family of floating-gate adapting synapses based upon transistor channel models," in *IEEE Int Symp Cir Syst*, vol. 1, 2004, pp. I–317.

[8] A. Basu, C. Petre, and P. Hasler, "Dynamics and bifurcations in a silicon neuron," *IEEE T Bio Cir Syst*, vol. 4, no. 5, pp. 320–328, 2010.

[9] F. Khoyratee, F. Grassia, S. Saïghi, and T. Levi, "Optimized real-time biomimetic neural network on FPGA for bio-hybridization," *Fnt Neurosc*, vol. 13, 2019.

[10] T. Levi, P. Bonifazi, P. Massobrio, and M. Chiappalone, "Editorial: Closed-loop systems for next-generation neuroprostheses," *Front Neurosc*, vol. 12, Feb. 2018.

[11] J. Hasler and H. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Fnt Neurosc*, vol. 7, 2013.

[12] S. Brink et al., "A learning-enabled neuron array IC based upon transistor channel models of biological phenomena," *IEEE T Bio Cir Syst*, vol. 7, no. 1, pp. 71–81, 2013.

[13] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J Physio*, vol. 117, no. 4, pp. 500–544, Aug. 1952.

[14] K. Yamazaki, V. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Br Sci*, vol. 12, no. 7, 2022.

[15] N. Brunel and M. van Rossum, "Quantitative investigations of electrical nerve excitation treated as polarization," *Bio Cybernet*, vol. 97, no. 5-6, pp. 341–349, Nov. 2007.

[16] E. Izhikevich, "Simple model of spiking neurons," *IEEE T Neural Net*, vol. 14, no. 6, pp. 1569–1572, 2003.

[17] J. Luo, G. Coapes, T. Mak, T. Yamazaki, C. Tin, and P. Degenaar, "Real-time simulation of passage-of-time encoding in cerebellum using a scalable FPGA-based system," *IEEE T Bio Cir Syst*, vol. 10, no. 3, pp. 742–753, 2016.

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

[19] S. Peron and F. Gabbiani, "Role of spike-frequency adaptation in shaping neuronal response to dynamic stimuli," *Bio Cybernet*, vol. 100, no. 6, pp. 505–520, Jun 2009.

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

[21] S. Yaghini Bonabi, H. Asgharian, S. Safari, and M. Nili Ahmadabadi, "FPGA implementation of a biological neural network based on the Hodgkin-Huxley neuron model," *Fnt Neurosc*, vol. 8, 2014.

[22] K. Akbarzadeh-Sherbaf, B. Abdoli, S. Safari, and A. Vahabie, "A scalable FPGA architecture for randomly connected networks of Hodgkin-Huxley neurons," *Fnt Neurosc*, vol. 12, 2018.

[23] F. Shama, S. Haghiri, and M. Imani, "FPGA realization of Hodgkin-Huxley neuronal model," *IEEE T Neural Syst Rehab Engr*, vol. 28, no. 5, pp. 1059–1068, 2020.

[24] X. Lin, X. Pi, X. Wang, P. Du, and H. Lu, "FPGA implementation of piecewise linear spiking neuron and simulation of cortical neurons," *Microproc Microsyst*, vol. 91, p. 104516, 2022.

[25] S. Haghiri, A. Naderi, B. Ghanbari, and A. Ahmadi, "High speed and low digital resources implementation of Hodgkin-Huxley neuronal model using base-2 functions," *IEEE Trans. Circuits Syst. I*, vol. 68, no. 1, pp. 275–287, 2021.

[26] S. Nitzsche, B. Pachideh, N. Luhn, and J. Becker, "Digital hardware implementation of optimized spiking neurons," in *Int Conf Neuromorph Comp*, 2021, pp. 126–134.

[27] C. Mead, *Analog VLSI and neural systems*, ser. Addison-Wesley VLSI systems series. Reading, Mass: Addison-Wesley, 1989.

[28] J. Kim and C. Fiorillo, "Theory of optimal balance predicts and explains the amplitude and decay time of synaptic inhibition," *Nature Comm*, vol. 8, no. 1, pp. 14 566–14 566, 2017.

[29] M. Versace, H. Ames, J. Léveillé, B. Fortenberry, and A. Gorchetchnikov, "Kinness: A modular framework for computational neuroscience," *Neuroin (Totowa, N.J.)*, vol. 6, no. 4, pp. 291–309, 2008.

[30] A. Natarajan and J. Hasler, "Implementation of synapses with hodgkin huxley neurons on the fpaa," in *IEEE Int Symp Cir Syst*, 2019, pp. 1–5.

[31] S. Gomar and A. Ahmadi, "Digital multiplierless implementation of biological adaptive-exponential neuron model," *IEEE T Cir Syst I*, vol. 61, no. 4, pp. 1206–1219, 2014.

[32] J. Hasler, "Starting framework for analog numerical analysis for energy-efficient computing," *J Low Power El Appl*, vol. 7, no. 3, 2017.

[33] B. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. Chandrasekaran, J. Bussat, R. Alvarez-Icaza, J. Arthur, P. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *P IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[34] C. Sherwood et al., "Invariant synapse density and neuronal connectivity scaling in primate neocortical evolution," *Cerebral Cortex (New York, N.Y. 1991)*, vol. 30, no. 10, pp. 5604–5615, 2020.

[35] M. Beyeler, E. Rounds, K. Carlson, N. Dutt, and J. Krichmar, "Neural correlates of sparse coding and dimensionality reduction," *PLoS comp. bio.*, vol. 15, no. 6, pp. e1 006 908–e1 006 908, 2019.

[36] M. Herrmann, J. Hertz, and A. Prügel-Bennett, "Analysis of synfire chains," *Network: Com Neu Syst*, vol. 6, no. 3, pp. 403–414, Jan. 1995.

[37] M. Gewaltig, M. Diesmann, and A. Aertsen, "Propagation of cortical synfire activity: survival probability in single trials and stability in the mean," *Neural Net*, vol. 14, no. 6, pp. 657–673, 2001.

[38] W. Maass, "On the computational power of winner-take-all," *Neural Comp.*, vol. 12, no. 11, pp. 2519–2535, 2000.

[39] S. Shah and J. Hasler, "SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier," *IEEE J Emer Sel Topics Cir Syst*, vol. 8, no. 1, pp. 28–37, 2018.

[40] P. Ferré, F. Mamalet, and S. Thorpe, "Unsupervised feature learning with winner-takes-all based STDP," *Fnt Comp Neurosc*, vol. 12, 2018.

[41] J. Lazzaro, S. Ryckebusch, M. Mahowald, and C. Mead, *Winner-Take-All Networks of O(N) Complexity*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, p. 703–711.

[42] A. Ghiasi and A. Zahedi, "Field-programmable gate arrays-based morris-lecar implementation using multiplierless digital approach and new divider-exponential modules," *Comp El Engr*, vol. 99, 2022.

[43] R. Fitzhugh, "Impulses and Physiological States in Theoretical Models of Nerve Membrane," *Biophys J*, vol. 1, no. 6, pp. 445–466, Jul. 1961.

[44] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon," *P IRE*, vol. 50, no. 10, pp. 2061–70, 1962.

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

[46] J. Hindmarsh and R. Rose, "A model of neuronal bursting using three coupled first order differential equations," *P R Soc Lond B Biol Sci*, vol. 221, no. 1222, pp. 87–102, Mar 1984.

[47] H. Wilson, "Simplified dynamics of human and mammalian neocortical neurons," *J Theo Bio*, vol. 200, no. 4, pp. 375–388, 1999.

[48] K. Woo, P. Chong, and L. Chian, "Implementation of parallel multiplications on FPGA," in *IEEE Ctrl Syst G R Coll*, 2015, pp. 32–37.

Swagat Bhattacharyya Swagat Bhattacharyya is a graduate research fellow in the School of Electrical and Computer Engineering at Georgia Institute of Technology researching bioinspired sensing and computing. In 2022, Swagat received a BSE in Electrical Engineering, a BS in Applied Physics, and a BS in Mathematics from Purdue University, the NSF Graduate Research Fellowship, and the Georgia Institute of Technology Presidential Fellowship.

Praveen Raj Ayyappan (Graduate Student Member, IEEE) received a B.Eng. in Electrical and Electronics Engineering from the University of Nottingham Malaysia Campus in 2022. He is now pursuing an M.S. and Ph.D. degree in Electrical and Computer Engineering in Georgia Institute Of Technology. His current research interests include NVM-based efficient large-scale analog computing, and designing neuromorphic circuits and systems.

Jennifer Hasler Jennifer Hasler is a full professor in the School of Electrical and Computer Engineering at Georgia Institute of Technology. Dr. Hasler received her M.S. and B.S.E. in Electrical Engineering from Arizona State University in 1991, received her Ph.D. from California Institute of Technology in Computation and Neural Systems in 1997, and received her Master of Divinity from Emory University in 2020. Dr. Hasler received the NSF CAREER Award in 2001, and the ONR YIP award in 2002. Dr. Hasler received the Paul Raphorst Best Paper Award, IEEE Electron Devices Society, 1997, a Best paper award at SCI 2001, Best Paper at CICC 2005, Best Sensor Track paper at ISCAS 2005, Best paper award at Ultrasound Symposium, 2006, and Best Demonstration paper award, ISCAS 2010.