Neural Spiking Dynamics in Asynchronous Digital Circuits

Nabil Imam, Kyle Wecker, Jonathan Tse, Robert Karmazin, and Rajit Manohar
Computer Systems Laboratory
Cornell University
Ithaca, NY, U.S.A.
{ni49,kyle,jon,rob,rajit}@csl.cornell.edu

***Abstract*—We implement a digital neuron in silicon using delay-insensitive asynchronous circuits. Our design numerically solves the Izhikevich equations with a fixed-point number representation, resulting in a compact and energy-efficient neuron with a variety of dynamical characteristics. A digital implementation results in stable, reliable and highly programmable circuits, while an asynchronous design style leads to energy-efficient clockless neurons and their networks that mimic the event-driven nature of biological nervous systems. In 65 nm CMOS technology at 1 V operating voltage and a 16-bit word length, our neuron can update its state 11,600 times per millisecond while consuming 0.5 nJ per update. The design occupies 29,500 μm^2 and can be used to construct dense neuromorphic systems. Our neuron exhibits the full repertoire of spiking features seen in biological neurons, resulting in a range of computational properties that can be used in artificial systems running neural-inspired algorithms, in neural prosthetic devices, and in accelerated brain simulations.**

I. Introduction

Neural circuits in the brain carry out complex computations involved in a variety of phenomena such as sensory perception, motor pattern generation, autonomous learning, and cognitive decision making. These computations are encoded in the spatiotemporal spiking activity of neurons in the system with different connectivity configurations between neurons with different dynamical properties implementing distinct algorithms. The human brain consists of approximately 10^{10} neurons and 10^{13} synapses, all contained within a volume of 2 L, weighing less than 4 lbs, and running on a 20 W power budget [1].

Artificial "neuromorphic" platforms intended to mimic the function of biological neural systems in real time and approach their compact size/weight and low power consumption, must capture the distributed nature of neural systems in an efficient way. Implementing large numbers of model neurons and synapses in a scalable and reliable platform and within aggressive area and power constraints requires careful design and implementation choices to be made [2]. Specialized hardware, especially Application Specific Integrated Circuits (ASICs), are effective tools with which these challenges can be met. ASIC "chips" offer better power and area efficiency than off-the-shelf processors as the datapath, memory, and communication circuits can be customized to implement high-density and large-scale neural networks.

Neuromorphic chips have a variety of potential applications such as in artificial perceptuo-motor systems, neural implants and brain simulation platforms. To implement the full range of spatiotemporal codes used in biological neural systems, neuromorphic hardware designs need to incorporate neuron models that reproduce the variety of spiking patterns of real neurons [3], and routing circuits that transmit information about the time and place of spikes across the system [4,5]. In practice, neuromorphic systems can be reconfigured to execute different algorithms by adjusting the dynamical properties of neurons (through configurable model parameters) and by changing the connectivity of the network (through configurable routing).

The response properties of neurons to synaptic inputs can be replicated through various mathematical models. These range from low-level multi-compartment models to high-level phenomenological models. Lower-level models precisely account for the morphological and electrochemical properties of neurons and synapses and capture their dynamical characteristics in detail. In contrast, higher-level models reduce the number of free parameters and capture only the neuron's essential properties at a more abstract level. Any reduction in model complexity translates directly to simplifications in ASIC circuit implementation and the corresponding savings in energy consumption and silicon area, making high-level neuron models attractive for implementing high-density and energy-efficient systems. Choosing an overly simple model is undesirable, however, as models must represent the wide range of dynamics used in neural computation.

One similarity between all biophysically-detailed models is that spiking dynamics is dictated by a *fast* inward activation current (typically the activation of Na$^+$ channels) and a *slow* outward recovery current (typically the inactivation of Na$^+$ channels and the activation of K$^+$ channels). Based on the phase plot of two variables affected by these currents, Izhikevich [6] reduced Hodgkin-Huxley type models into a two-dimensional system that captured the subthreshold and spike-initiation dynamics of a neuron while compromising the precise shape of its spike. The model uses the interaction between a fast variable, v, and a slow variable, u, to produce the various spiking patterns observed in cortical neurons within 14 arithmetic operations [3]. Models such as the Izhikevich model that can capture a neuron's important properties without accounting for all the biological details is ideally suited for dense and energy-efficient neuromorphic systems.

The Izhikevich neuron model is formulated as a system of two differential equations

$$v' = ev^2 + fv + g - u + I \tag{1}$$

$$u' = a(bv - u) \tag{2}$$

with an after-spike reset.

$$\text{if } v > \theta, \text{ then: } v = c \text{ and } u = u + d \tag{3}$$

v' and u' are the time derivatives of v and u. The parameters e, f, and g are usually fixed to 0.04, 5, and 140, respectively, although other values are sometimes preferred [7]. The parameter a describes the speed of the recovery (slow) variable, b describes the influence of the activation (fast) variable on the recovery variable, and c and d describe the reset value of the variables after a spike. By tuning these four parameters, different dynamical characteristics can be generated.

In this paper we present an asynchronous Izhikevich neuron implemented in 65 nm CMOS technology for use in digital neuromorphic chips. Asynchronous circuits naturally perform event-driven computation, minimizing power consumption in the absence of data. We gain additional energy and area savings by representing the Izhikevich equations using fixed- instead of floating-point numbers. While this results in a loss of precision, our fixed-point implementation is capable of producing the same spiking patterns as a floating-point implementation. We illustrate how our circuit's speed, energy consumption and area changes with different levels of precision. With a word length of 16 bits, our neuron occupies 29,500 μm^2 and updates the Izhikevich equations 11,600 times per millisecond while consuming 0.5 nJ per update, making it the most compact and energy-efficient two-variable digital neuron we are aware of to date. Our asynchronous neuron interfaces seamlessly with the event-driven routing fabrics typically used in neuromorphic systems, removing the overhead associated with network interface and synchronization circuitry.

II. Design Considerations

Large-scale implementations of Izhikevich neurons have been previously implemented in software, using computing clusters consisting of high-performance/high-power-consuming general purpose processors [12], lower power fixed-point processors [13], FPGAs [14] and GPUs [15].

General-purpose processors are based on a von Neumann computing architecture, wherein processing and memory are physically separated and instructions are largely executed following a sequential compute model. This architecture is not a natural fit to the parallel and event-driven nature of neural computations. The storage and retrieval of a large number of neuron and synapse parameters from off-chip memory arrays lead to high energy consumption and increased latency, limiting efficiency and scalability. To achieve large-scale implementations in software requires supercomputer-levels of computational power and the associated costs, for example [16].

Custom ASICs, as previously discussed, present an efficient design point for implementing large-scale brain-like networks. Neuron models can be implemented using compact and energy-efficient circuits [2], memories storing configuration parameters can be tightly coupled to computing datapaths [17], routing fabrics can be tailor-made for spike-based communications [18], and synchronization methods can be deployed to maintain hardware correspondence with software [19].

In an effort to minimize transistor count, designers can make use of analog circuit implementations of neurons. Transistor transconductance properties can be used to mimic the operation of neurons, reducing the number of transistors required for reproducing the dynamics of individual neurons [8] in comparison to digital implementations. Previous work has shown the implementation of Izhikevich-like equations in analog circuits [9-11], resulting in a number of spiking patterns.

These implementations, although compact and low-power, have a number of drawbacks characteristic of analog circuits. Device mismatch and environmental fluctuations, e.g. changes in ambient temperature or voltage, limits correspondence between intended hardware configuration, i.e. the desired neural algorithm, and the actual circuit operation. This non-determinism can adversely affect the usability of such circuits in real-world applications and simulation platforms. In addition, analog circuits do not scale well to deep-submicron CMOS processes [25], in part due to a lack of high-density capacitors and increasing sub-threshold currents.

In contrast, digital circuits offer a deterministic and high-precision platform. By leveraging the digital abstraction, neural algorithms can be guaranteed to reliably operate precisely as specified. This is particularly important in cases where one-to-one correspondence is expected with a software model. By taking full advantage of CMOS technology scaling, the power and area overheads associated with a digital implementation can be overcome. For these reasons our neuron is a digital circuit implementation of the Izhikevich model.

We make use of our asynchronous digital circuit design methodology [22], which transforms a high-level program description into a set of parallel CMOS circuits that communicate with one another via delay-insensitive channels. Our circuits satisfy the Quasi-Delay Insensitive (QDI) model, which has minimal timing assumptions. As a result, QDI circuits are robust to timing, temperature, voltage, and device variations. This hardware model closely emulates that of a biological neural system, where neurons communicate with one another via chemical and electrical signaling often without a global synchronization signal or "clock", as would be the case in a synchronous digital system. Any synchronization in our circuits is accomplished locally between hardware processes through request-acknowledge handshakes.

An asynchronous design has the advantage of demand-driven switching activity, consuming dynamic energy only when computation is taking place [21]. While synchronous systems can implement this by adding clock-gating circuits, asynchronous circuits naturally provide the benefits of fine-grained clock gating with no additional overhead. Removing the clock entirely also has the benefit of eliminating the clock distribution circuitry and the associated area and power overheads, which can be very expensive for large neuromorphic systems.

Globally Asynchronous, Locally Synchronous (GALS) architectures have been the traditional compromise for large synchronous systems when faced with the difficulties of clock distribution. However, large-scale neuromorphic systems are already globally asynchronous. In these systems, it is typical to use clockless event-driven routing fabrics to communicate the sparse and irregular spikes between neurons [5]. As our neuron is itself self-timed, i.e. asynchronous, it can seamlessly integrate with the routing fabric without the necessity to cross timing domains as would be the case in a GALS system. Applying an asynchronous design methodology across the whole system eliminates the need for synchronization circuits and design difficulties such as timing closure [26].

Furthermore, because they are self-timed, asynchronous circuits are robust to delay variations resulting from CMOS process variations and environmental changes to ambient temperature, system voltage, or electrical noise sources [27]. This robustness, especially compared against analog circuits, directly translates to preserving neuromorphic system functionality across a wider array of situations, such as the unusual operating environments found in robotics applications.

While these benefits typically come at the cost of increased design effort, developments in asynchronous design methodologies [22] and circuit synthesis tools [28] have streamlined the construction of asynchronous circuits from high-level descriptions to physical layout.

III. Neuromorphic System Architecture

Neuromorphic systems are made up of many neurons connected by synapses. Typical neurons have 10^3 to 10^4 synapses, which make point-to-point connections with dedicated wires on an ASIC intractable for any significant number of neurons. Fortunately, neuron activity is measured in the Hz range, whereas ASIC wire bandwidth can reach hundreds of MHz to several GHz. Thus, most custom-ASIC neuromorphic systems [19,23,24] time multiplex wire usage to implement the connections of a group of neurons, encapsulated in *neural cores*.

Communication in neuromorphic systems is discretized into address-event representation (AER) packets [4], which encode information about neuron spiking activity. The minimal packet is a *(source, destination, timestamp)* tuple indicating the source, destination, and time of a spike. Inter-core packet traffic is handled by routing circuits [5,18] using the source/destination packet fields for addressing. As an example, we show a 2D mesh routing network in Fig. 1.

Neurons are typically organized as illustrated in Fig. 2. When exiting or entering a core, AER packets traverse a lookup table that maps the source and destination core fields of the packet to individual neurons in the local core. These lookup tables are indicated as the synaptic memory and routing memory in Fig. 2.

As an arriving packet traverses the synaptic memory, its source field is mapped to a user-configured subset of neurons in the neuron array, which are the local spike recipients. Any

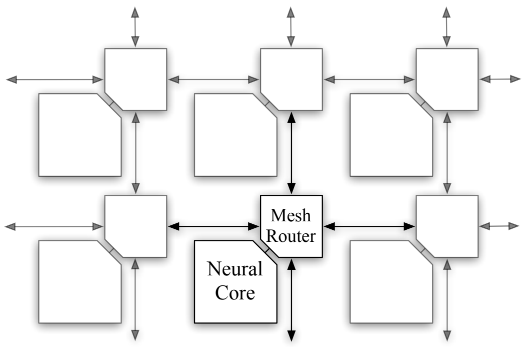

Fig. 1. Array of Neural Cores, connected by a 2D Manhattan mesh network. Other network topologies are possible [18].

transforms based on the chosen synapse model may be applied here, or in the circuitry of individual neurons. Typically, a weight parameter, stored in the synaptic memory, is applied to the spike before being forwarded to the appropriate neurons. In some neuromorphic cores, the timestep field of the AER packet may be used to delay the delivery of the spike to the target neurons to model the axonal delay of a biological neuron. In the outbound packet case, as a locally generated spike leaves the core, the routing memory maps the locally spiking neuron to a remote core, writing the appropriate remote core to the destination field. High neural fanout can be efficiently implemented by clustering neurons that are often co-recipients of a spike, in the same core.

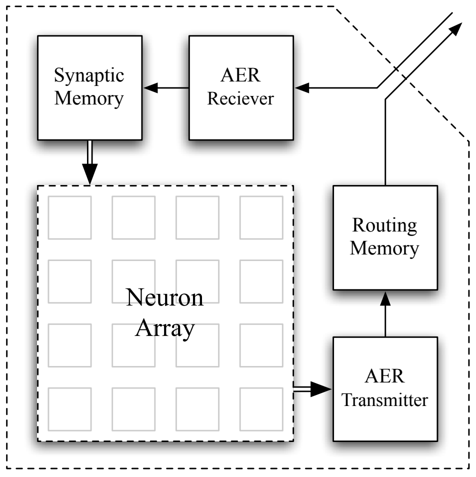

Fig. 2. A Typical Neural Core.

Speed, energy efficiency, and area, all of which affect the scalability of the system, are primarily determined by the following architectural structures:

- **Neuron**: The speed of neural updates governs how fast the entire system can run. The energy per neuron update, summed across all neurons, contributes significantly to overall system energy consumption. The area footprint of a neuron dictates the size of the neuron array.

- **AER TX/RX**: The speed of the AER transmitter/receiver determines the maximum number of neuron communications that can be multiplexed together, setting a direct limit on the size of the neuron array. Spiking patterns are typically bursty and sparse. Therefore, event-driven asynchronous AER circuits are usually used to minimize power consumption during idle periods while maintaining high throughput during bursty periods.
- **Routing Network**: The network must have sufficient bandwidth to accommodate spiking activity from all the cores in the system, while guaranteeing packet delivery within the axonal delay time. Again, asynchronous circuits are usually used to handle bursty and sparse spiking traffic efficiently.
- **Synapses**: Synapses outnumber neurons by three to four orders of magnitude, so the design of the synaptic memory is a large determinant of the neural core area for an on-chip memory design. This memory will critically affect the energy consumption and speed of operation when off-chip memory is employed.

These systems can be designed to run a discrete-time simulation based on an externally defined timestep. In such a design, the total synaptic current for each update (I in Equation 1) is passed to the neuron by circuits between the neuron and the synaptic memory, that accumulate the spikes coming in for one timestep. A global update signal oscillating with a period equal to the timestep of the simulation initiates a neural update. The period of the update signal provides an absolute bound within which all the communication and computation in the system has to take place. With the use of synchronization circuits, this kind of design can create a one-to-one correspondence between hardware operation and software simulation [17,19].

Alternatively, these systems can run freely without a global update signal. Neural updates can be carried out as AER packets come in, or updates can be continuously made and incoming packets checked for between updates. Such a design would operate at maximum speed, but will not produce a strict correspondence between hardware and software due to a lack of synchronization between the neurons and the non-deterministic routing network.

IV. Neuron Design and Implementation

In designing a neuron, we assume a typical system architecture with neurons operating next to a synaptic memory array, as described in Section III. We optimize primarily for neuron area to maximize the number of neurons in each core, and keep energy and speed as an important secondary and tertiary considerations respectively.

Minimizing the number of arithmetic units needed to implement the Izhikevich equations results in small circuit footprint. Therefore, we break Eqs. 1 and 2 into the sequential steps shown in Table I.

Each row in Table I corresponds to an add and/or multiply. Each step uses the results of one of the previous steps. The addition of the lump synaptic current, I, is done in the last step in order to parallelize the accumulation of synaptic currents to the neuron with the other steps (1-6). All computation takes place on fixed-point numbers represented in two's complement. We tested our circuit with different levels of precision in the fixed-point numbers, discussed in Section V.

TABLE I
Calculation Breakdown

Step	Multiplier	Adder
1	ev	$g-u$
2	bv	$ev+f$
3	$v(ev+f)$	$bv-u$
4	$a(bv-u)$	$v(ev+f)+g-u$
5		$v+v(ev+f)+g-u$
6		$u+a(bv-u)$
7		$v+v(ev+f)+g-u+I$

This formulation requires a datapath of two arithmetic computation units, a two-operand fixed-point adder and a two-operand fixed-point multiplier, as shown in Fig. 3. A control block, detailed in Section IV-A, feeds the datapath with the appropriate operands and routes the outputs to the appropriate place. The neuron state variables (i.e. v and u) are stored in a local neuron memory along with the neuron parameters.

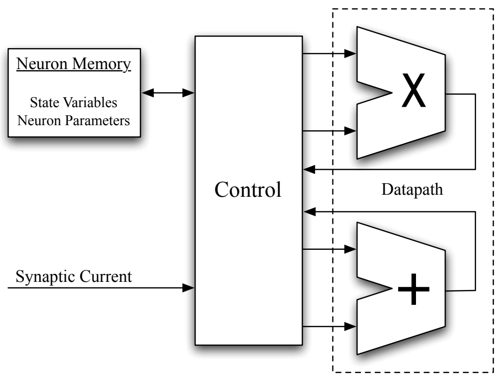

Fig. 3. Neuron Architecture

A. Control

The control of the neuron is illustrated in Fig. 4. A finite state machine, implemented using an asynchronous token-ring [20], controls a series of merges (M) and splits (S) which route operands to/from the adder and multiplier to implement the steps of Table I. Operands can be sourced from either the neuron memory or from datapath results. The results of the multiplier are routed to the adder and the results of the adder can be routed to the multiplier, back to the input of the adder, to a comparator, or back to registers in the local neuron memory. The comparator evaluates if v has crossed threshold (usually set to 30), the result of which decides if v and u are to be reset as in Eq. 3 and if a spike is to be sent to the AER transmitter.

B. Datapath

The datapath is made of a fixed-point adder and a fixed-point multiplier. The adder is implemented as a simple ripple-

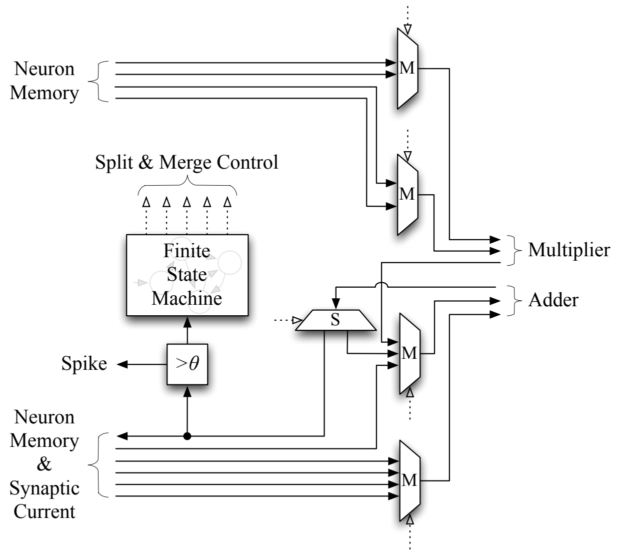

Fig. 4. Neuron Control Block

carry design to minimize area costs. The multiplier is a Booth-encoded serial multiplier employing a counterflow structure [29], where control and data flow in opposite directions in a pipeline. A high-level representation can be seen in Fig. 5.

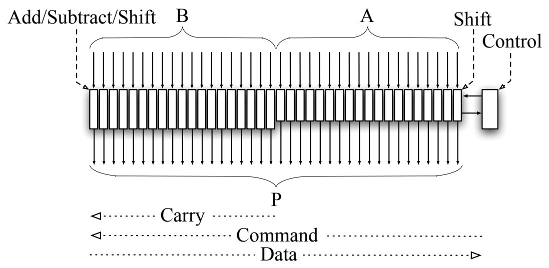

Fig. 5. Counterflow Serial Booth-Encoded Multiplier

Each unit in the pipeline of Fig. 5 is either a shifter block (labeled as "Shift") or an ALU block (labeled as "Add/Subtract/Shift"). A multiplier control resides on the right-most side of the pipeline. The shifters initialize with the bits of the multiplier A, and the ALUs initialize with the bits of the multiplicand B. The pipeline then shifts and processes the values of A and B in a bit-serial fashion, depending on booth commands sent from the multiplier control unit. This unit determines the next command to be injected into the pipeline by analyzing two consecutive data bits coming out of the rightmost shifter. Based on the Booth algorithm, an addition command is sent if a 1 is followed by a 0, a subtraction command is sent if a 0 is followed by a 1, and a shift command is sent if consecutive bits match (00 or 11). A sentinal token initializes in the middle of the pipeline (in the rightmost ALU) and flows with the data. When the control block receives this token, a final command is sent causing each unit to pass its value out to P.

The ALU unit of the pipeline is show in Fig. 6. The blocks labeled "PCHB" and "WCHB" are specific kinds of asynchronous buffers [30], that copy the data on their input channel to their output channel. The "en" label in the WCHB blocks refers to an enable signal. Depending on the command ($CMDin$) coming from the right, any one of the four WCHBs is enabled to produce the data value going to the right (D_{out}). The input C_{in} and the output C_{out} are the carry-in and carry-out bits to the neighboring units of the pipeline. The shifter units are similar to Fig. 6, but they don't have the add/sub blocks and the associated WCHBs and carry bits. Also, the input to the initialization buffer (the top-most WCHB in Fig. 6) of a shifter is a bit from the multiplier A.

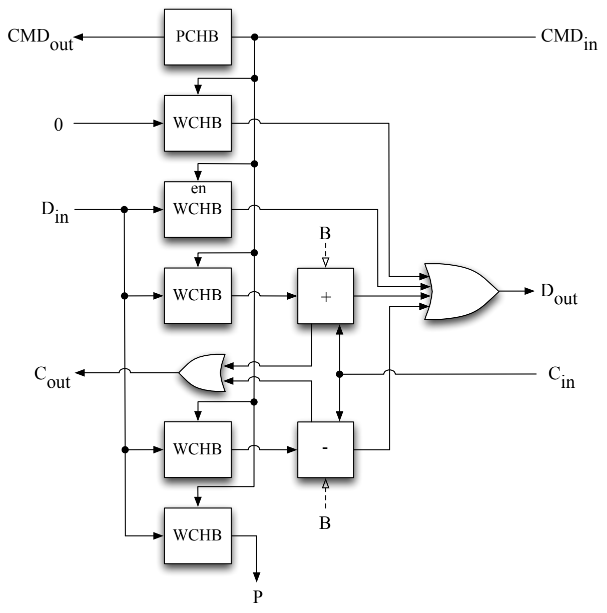

Fig. 6. Multiplier Add/Subtract/Shift Unit

The multiplier block is the most significant driver of the area, energy consumption and speed of our neuron. We used the Booth algorithm because it leverages strings of consecutive zeros or consecutive ones in the multiplier (A) to reduce the number of additions and subtractions required in a standard bit-serial algorithm. Our asynchronous design methodology is well-suited for the counterflow pipeline implementation of the algorithm, and leads to an area and energy-efficient design. The counterflow architecture also enables overlapped execution of consecutive multiplications since units in the pipeline can start processing the next set of operands without waiting for all other units to finish.

We used a bit-serial multiplier to achieve a compact implementation. Alternatively, an array-based multiplier can be used. This will lead to better energy efficiency and higher speed at the cost of larger area.

V. Results

We transformed the processes of a single neuron in the *neuron array* of Fig. 2 into QDI CMOS circuits using the synthesis technique described in Section II. We evaluated our design in a commercially available 65 nm CMOS process. We present SPICE simulation projections for the process at 25°C in the Typical-Typical corner at 1 V. Our SPICE framework includes conservative wire capacitance estimates, resulting in performance and power numbers similar to actual silicon measurements from a variety of fabricated chips [31]. We estimate area via an automated tool that synthesizes "standard" cells for place and route [28]. On average, our tool produces layout 2x larger than unassisted human layout.

Our choice of a fixed-point number representation results in an accumulation of round-off errors after every arithmetic operation. In time, the results of a fixed-point implementation will drift away from the results of a floating-point one. The lower the fixed-point precision, the larger the fixed- to floating-point discrepancy, as shown in Fig. 7. In this figure, a (x,y) fixed-point number denotes a value with x bits of precision in the integer and y bits of precision in the decimal. Despite the discrepancy, neural computations can be carried out in low precision implementations using the temporal patterns of spikes irrespective of how closely the fixed- and floating-point solutions match (see the caption of Fig. 8 for further description). Precise temporal characteristics of these patterns, such as the onset latency or the inter-spike interval, can be adjusted by tuning the parameters of the model. Higher precision implementations will allow a higher dynamic range, and therefore more flexibility, in the adjustments of these values.

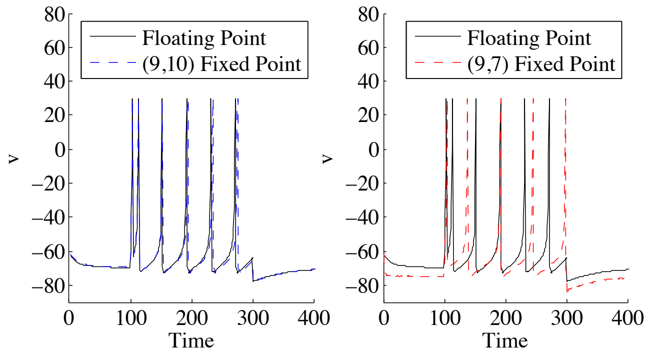

Fig. 7. Fixed- versus floating-point implementations with parameters for tonic spikes. Step current is applied from time 100 to 300. The v and time axes are dimensionless.

Several different spike patterns generated by our circuit along with the associated phase portraits is illustrated in Fig. 8. Each of the patterns can be used in distinct computations, as described in the caption. Our neuron reproduces all the prominent spiking patterns seen in biological neurons [3]. Many of them can be generated through a (9,7) fixed-point two's complement number representation. Some patterns, such as the resonator in Fig. 8, require 10 bits of precision in the fraction.

The results of our SPICE simulations for various fixed-point precisions are presented in Table II. Here, the number of bits in the integer is fixed at 9, and that in the fraction is varied. With a (9,7) fixed-point representation (16 bits), our circuit performs over 11,600 updates per millisecond, consumes 0.5 nJ per update and occupies 29,500 μm^2. To our knowledge, this is the most energy- and area-efficient implementation of a two-variable digital neuron reported to date.

TABLE II
THE EFFECTS OF PRECISION AT 1 V

Precision [b]	Update Speed [ns]	Energy/Update [nJ]	Area [μm^2]
12	64	0.289	22,505
16	86	0.511	29,500
20	107	0.776	36,862
24	129	1.109	44,233

Table III lists the performance and energy consumption of our circuit for different supply voltages. Our neuron can carry out over 18,500 updates per millisecond at 1.2 V, or consume only 313 pJ of energy per update at 0.8 V.

TABLE III
SUPPLY VOLTAGE SCALING WITH 16-BIT PRECISION

Voltage [V]	Update Speed [ns]	Energy/Update [nJ]
1.2	54	0.726
1.1	68	0.590
1.0	86	0.511
0.9	123	0.403
0.8	202	0.313

The static power dissipation of a 16-bit instantiation of our neuron at 1 V is 203 nW. While this can be further reduced with the use of high-Vt transistors and static power reduction techniques, the synapse memory rather than the neuron circuits will dominate the leakage current of a large system since the former exceeds the latter by three to four orders of magnitude.

VI. DISCUSSION

Our neuron is intended for dense digital neuromorphic systems. A 16-bit implementation of our circuit occupies less than 30,000 μm^2 (Table II) in 65 nm technology. Hundreds of thousands of these neurons can be packed in a medium- to large-sized chip. Digital circuits are presently being fabricated using feature sizes as small as 22 nm. By leveraging this scaling, digital models of large brains can be constructed using our neurons in compact multi-chip systems.

As discussed in Section II, memory organization significantly affects the speed, energy-efficiency, and scalability of neuromorphic systems. Our design is well suited for architectures that exclusively use local on-chip memory, such as the recently implemented neurosynaptic core [17,19]. In the neurosynaptic core architecture, individual neurons store their parameters in local registers and a crossbar memory local to a neuron core implement large synaptic fanout efficiently. A system in which our energy-efficient neurons are embedded in this architecture can achieve energy-efficiency two or three orders of magnitude better than software implementations.

A 16-bit instantiation of our circuit operates 11,600 times faster than real time (1 ms timestep) at 1 V (Table III). Systems that are designed to operate at lower speeds, such as

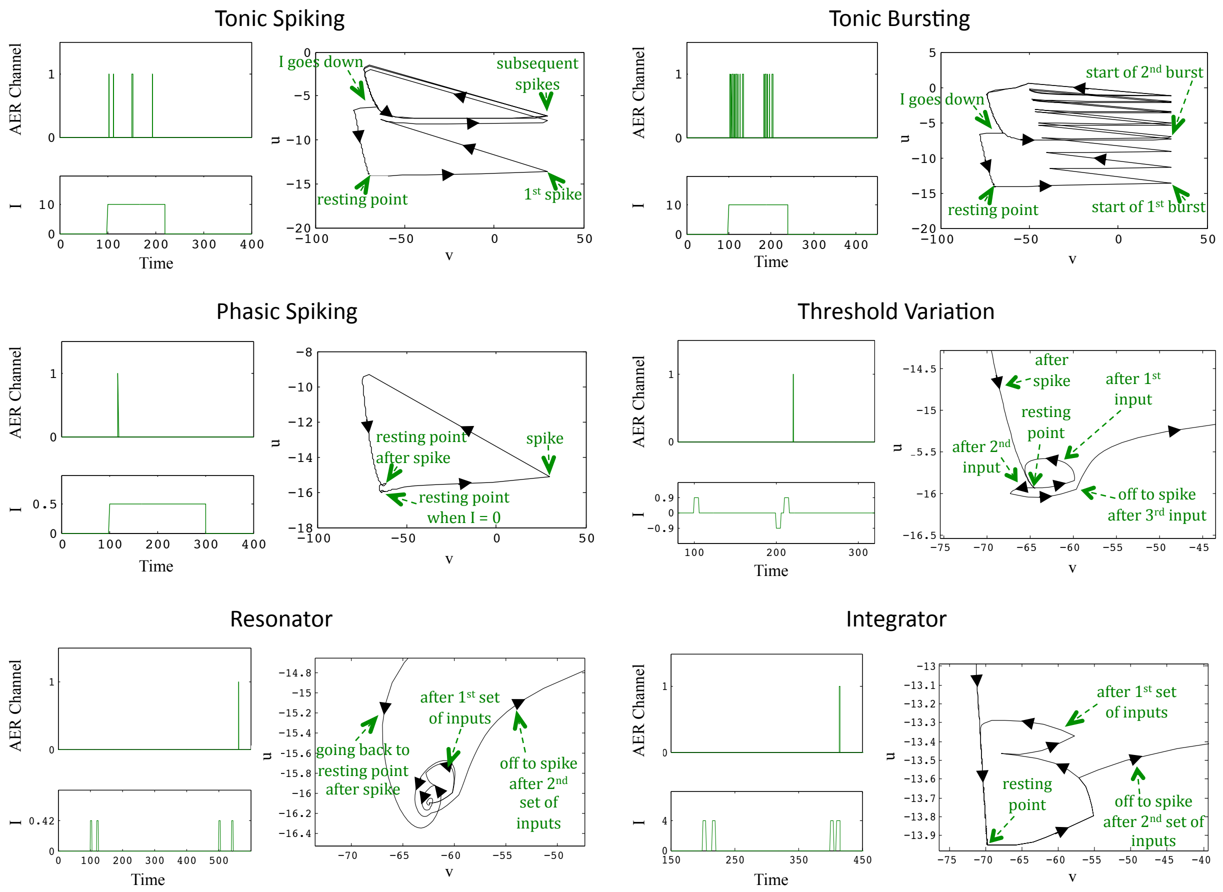

Fig. 8. Neural Dynamics in Asynchronous Circuits: A subset of spike patterns generated by our circuit from transistor-level simulations. For each pattern, the output generated in the spike channel of the neuron, the stimulus current, and the phase portrait are shown. The phase portrait illustrates the interaction between the fast variable v and the slow variable u that generate neural dynamics. Each pattern can be used for specific computational tasks. Tonic spikes can indicate the consistent presence of a stimuli, analogous for example to P ganglion cells in the retina. In contrast, a phasic spike can indicate just the start of a stimulus like M ganglion cells in the retina do. Resonators can implement frequency-modulated interactions such as those found in the cochlear nucleus, while integrators can perform coincidence detection similar to sound-localizing neurons in the auditory midbrain. Neurons exhibiting threshold variation can be modulated in a variety of ways by precise interactions between inhibitory and excitatory synapses, while bursting neurons are believed to be behind the gamma-frequency oscillations that are hypothesized to perform various synchronization roles in the brain.

a platform intended for real-time operation, can get significant improvements in neuron density by using one physical neuron circuit to implement multiple time-multiplexed virtual neurons, as previously demonstrated in a FPGA implementation [14]. Instead of storing the parameters of neurons in separate registers, a combined neuron memory array along with a single instance of our circuit can implement several hundred neuron updates. This of course comes at an expense of extra energy consumption associated with moving parameters to and from a large shared neuron memory.

Given the sparse activity rates in neural networks, a neuron may not receive synaptic input for long periods of time. In the majority of such instances, the neuron will sit idle at a fixed point in its phase plane. In these cases, carrying out the updates of Table I will lead to wasteful energy consumption. An addition to the neuron control block of Fig. 4 can be made to prevent this energy overhead at a slight cost of an area overhead. A sampling block, that periodically samples both the state of the incoming synaptic current and the state of the variables v and u, can detect when updates should be halted. When the synaptic current stays zero for some period of time, and if in that period both v and u are always within some small region around the same point in the phase plane, the control can shut down neural updates to save energy until a synaptic current comes in.

Our neuron can be used in several neuromorphic applications. These applications employ algorithms inspired from neural activity in the brain to solve a variety of artificial sensory-motor problems where traditional computer algorithms have fallen short. These include for example, visual, auditory and chemical signal processing [32,33], and locomotor central pattern generation [34]. The variety of computations used here will benefit from using the full repertoire of dynamical features that our neuron exhibits, and our compact and energy-efficient circuits will allow these algorithms to be deployed in real-world applications.

Another application of our circuits is in brain-embodied robots [35]. These robots explicitly model the brain's interaction with the body and the environment in real time. They are promising tools for studying the brain and offer novel ways of designing intelligent robots. In contrast to robots based on conventional artificial intelligence, these brain-based devices autonomously learn from their experience without a priori instructions. The activity of *all* elements in the neural circuits of the devices can be recorded and examined in detail while they interact with the body and the environment, providing critical insights into brain function not achievable through current techniques in animal experimentation. Fast, energy-efficient and compact circuits such as ours are crucial in developing these platforms.

Brain-machine interfaces have seen rapid development over the past decade [36], and it's another area where our circuits will be of use. These interfaces are intended for neural prosthetics, and create a direct communication channel between biological neurons and artificial systems. They translate raw neuronal signals into motor commands and provide sensory feedback to the brain. This is an area where both analog and digital silicon neuron implementations may be useful, with ultra-low-power analog neurons at the interface of biology and silicon and reliable, higher-precision digital neurons performing computations in the rest of the system.

Finally, our circuits can also be deployed in brain simulation technologies. With data pouring in from anatomical and physiological investigations of brain structure and function, several research groups are attempting to integrate the information through large-scale computer simulations of neural systems [12,16]. Simulating a system as complex as the brain requires high-performance computing platforms, and our circuits can be efficient building blocks of such simulators.

In conclusion, we have designed the most compact and energy-efficient two-variable digital neuron to date. We have demonstrated a wide range of dynamical properties that our asynchronous neuron exhibits. Our circuits can operate in real time or faster, and naturally fit into the event-driven communication fabrics of neuromorphic architectures. Our neuron is ideally suited as a building block for a variety of real-time, compact and low-power neuromorphic systems.

Acknowledgments

The authors thank the 2012 Neuromorphic Cognition Engineering Workshop for discussions about digital neurons.

References

[1] S. Herculano-Houzel. "The human brain in numbers: a linearly scaled-up primate brain." Frontiers in Human Neuroscience 3 (2009).

[2] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, no. 00073, 2011.

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

[4] N. Imam and R. Manohar, "Address-event communication using token-ring mutual exclusion," in *ASYNC*, pp. 99–108, IEEE Computer Society, 2011.

[5] P. Merolla, J. Arthur, B. Shi, and K. Boahen, "Expandable networks for neuromorphic chips," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, no. 2, pp. 301–311, 2007.

[6] E. Izhikevich, *Dynamical systems in neuroscience: the geometry of excitability and bursting*. MIT press, 2006.

[7] E. Izhikevich, "Simple model of spiking neurons," *Neural Networks, IEEE Transactions on*, vol. 14, no. 6, pp. 1569–1572, 2003.

[8] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, pp. 1629 –1636, Oct. 1990.

[9] J. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour," *Neural Networks*, vol. 21, no. 2, pp. 524–534, 2008.

[10] V. Rangan, et al. "A subthreshold avlsi implementation of the izhikevich simple neuron model," in *EMBC*, pp. 4164–4167, IEEE, 2010.

[11] A. van Schaik, et al. "A log-domain implementation of the izhikevich neuron model," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 4253–4256, IEEE, 2010.

[12] E. Izhikevich and G. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the national academy of sciences*, vol. 105, no. 9, pp. 3593–3598, 2008.

[13] T. Sharp, et al. "Power-efficient simulation of detailed cortical microcircuits on SpiNNaker." Journal of Neuroscience Methods (2012).

[14] A. Cassidy and A. Andreou, "Dynamical digital silicon neurons," in *BioCAS 2008. IEEE*, pp. 289–292, IEEE, 2008.

[15] J. Nageswaran, et al. "Efficient simulation of large-scale spiking neural networks using cuda graphics processors," in *IJCNN*, pp. 2145–2152, IEEE, 2009.

[16] R. Ananthanarayanan, et al. "The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses." Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. ACM, 2009.

[17] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm," *CICC 2011*, September 2011.

[18] D. Vainbrand and R. Ginosar. "Scalable network-on-chip architecture for configurable neural networks." *Microprocessors and Microsystems 35.2 (2011):* 152-166.

[19] N. Imam, F. Akopyan, J. Arthur, P. Merolla, R. Manohar, and D. Modha, "A digital neurosynaptic core using event-driven qdi circuits," in *ASYNC 2012*, Pp. 25–32, IEEE, 2012.

[20] A. J. Martin, "Distributed mutual exclusion on a ring of processes," *Sci. Comput. Program.*, vol. 5, pp. 265276, October 1985.

[21] R. Manohar, "A case for asynchronous computer architecture," *Proceedings of the ISCA Workshop on Complexity-Effective Design*, June 2000.

[22] A. J. Martin, "Programming in VLSI: From communicating processes to delay-insensitive circuits," in *Developments in Concurrency and Communication, UT Year of Programming Series* (C. A. R. Hoare, ed.), pp. 1–64, Addison-Wesley, 1990.

[23] K. Boahen, "Neurogrid: emulating a million neurons in the cortex," *EMBC*, IEEE, 2006.

[24] S. Joshi, et al. "Scalable event routing in hierarchical neural array architecture with global synaptic connectivity," in *CNNA 2010*, pp. 1–6, IEEE, 2010.

[25] A. J. Annema, B. Nauta, R. van Langevelde, and H. Tuinhout. "Analog circuits in ultra-deep-submicron CMOS" IEEE SSC, vol. 40

[26] A. J. Martin and M. Nystrom, "Asynchronous Techniques for System-on-Chip Design,". IEEE, vol. 94, no. 6, pp. 1089–1120, Jun. 2006.

[27] D. Fang, J. Teifel, and R. Manohar. "A high-performance asynchronous FPGA: Test results." Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on. IEEE, 2005.

[28] R. Karmazin, C. Ortero, and R. Manohar. "cellTK: Automated Layout for Asynchronous Circuits with Nonstandard Cells." To Appear In ASYNC 2013.

[29] J. Hensley, A. Lastra, and M. Singh, "An area- and energy-efficient asynchronous Booth multiplier for mobile devices," ICCD 2004.

[30] A.M. Lines. "Pipelined asynchronous circuits." (1998).

[31] http://vlsi.cornell.edu/chips.php

[32] S. Liu and T. Delbruck, "Neuromorphic sensory systems." Current opinion in neurobiology" 20.3 (2010): 288.

[33] N. Imam, et al. "Implementation of olfactory bulb glomerular-layer computations in a digital neurosynaptic core." Frontiers in Neuroscience 6 (2012).

[34] A.J. Ijspeert. "2008 Special Issue: Central pattern generators for locomotion control in animals and robots: A review." Neural Networks 21.4 (2008): 642-653.

[35] G.M. Edelman. "Learning in and from brain-based devices." Science 318.5853 (2007): 1103-1105.

[36] M.A. Lebedev and M.A. Nicolelis. "Brain-machine interfaces: past, present and future." TRENDS in Neurosciences 29.9 (2006): 536-546.