A Learning-Enabled Neuron Array IC Based Upon Transistor Channel Models of Biological Phenomena

Stephen Brink, Stephen Nease, Paul Hasler, *Senior Member, IEEE*, Shubha Ramakrishnan, Richard Wunderlich, Arindam Basu, *Member, IEEE*, and Brian Degnan

***Abstract*—We present a single-chip array of 100 biologically-based electronic neuron models interconnected to each other and the outside environment through 30,000 synapses. The chip was fabricated in a standard 350 nm CMOS IC process. Our approach used dense circuit models of synaptic behavior, including biological computation and learning, as well as transistor channel models. We use Address-Event Representation (AER) spike communication for inputs and outputs to this IC. We present the IC architecture and infrastructure, including IC chip, configuration tools, and testing platform. We present measurement of small network of neurons, measurement of STDP neuron dynamics, and measurement from a compiled spiking neuron WTA topology, all compiled into this IC.**

***Index Terms*—Electrical implementation of neurobiology, neuromorphic engineering.**

I. Overview of a Single IC Neuron Learning Array

MULTIPLE research efforts have been looking for an array of neuron elements with realistic biological dynamics at a density that enables looking at neural dynamics of 100 neurons or more. A biological neuron is defined by its Soma, dendrite, synapses, and axons, as seen in Fig. 1(a). For our electrical IC models, we will follow a similar block diagram for the basic components. Incoming axon lines form a connection through synapses to the neuron dendrite line that feeds into the soma block of the neuron. The soma block creates the dynamics/computation to send a resulting action potential, often described as an event, to its output axon connection. For this discussion, we will assume the model of the dendrites is a wire, typical of most modeling and implementation approaches; handing dendritic computation is beyond the scope of this work and discussed elsewhere [1], [2]. Previous efforts have achieved part of these solutions [3]–[5], often having a tradeoff between dense circuit structures or modeling biological behavior.

Our goal was to build an IC that implements multiple neurons that uses biologically realistic transistor based models of neurobiological computation. Previous work has shown dense, biologically relevant circuit models of synaptic behavior, including biological computation [7] and biological learning dynamics [8], as well as transistor channel models of biological channels enabling soma (as well as dendritic) dynamics [9]. Fig. 1(a) shows this paper's viewpoint of implementing a silicon neuron, enabling arrays of neurons on a single IC. Our approach uses these component level innovations to build a 100 neuron, 30,000 synapse chip in 350 nm double-poly CMOS process in die area of 5 mm × 5 mm. We show a die photo of the fabricated IC in Fig. 1(b). The synaptic array consumed roughly 3 mm^2 in area; therefore we can imagine in a single retical size chip in the same process to enable an array of one million synapses and thousands of neurons on a single IC, with these numbers only increasing substantially as we move to modern IC processes.

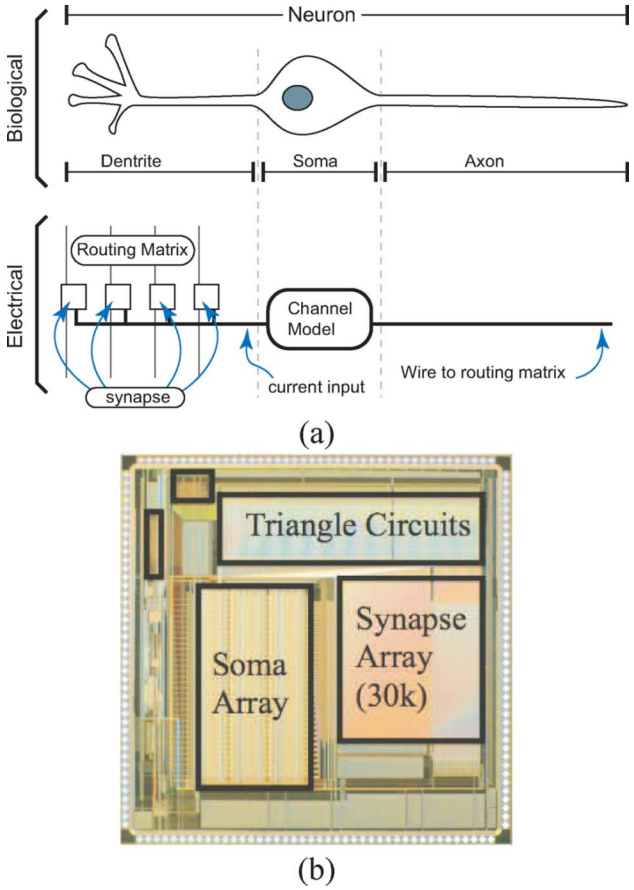

Fig. 1. Our goal was to build an IC that implements multiple neurons that uses biologically realistic transistor based models of neurobiological computation. A biological neuron is defined by its soma, dendrite, synapses, and axons. (a) For our electrical IC models, we will follow a similar block diagram for the basic components, including efficient models of synapses, channel regions in the soma, and communication of spikes to other synapses. For this work, we will assume the typical model that dendrites can be simply modeled as wires. (b) Die Photo of the IC of 100 neurons and 30,000 synapses consumes 5 mm × 5 mm area in 0.35 μm CMOS process.

Manuscript received November 20, 2011; revised February 07, 2012; accepted March 28, 2012. This paper was recommended by Associate Editor J. Van der Spiegel.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-250 USA (e-mail: phasler@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBCAS.2012.2197858

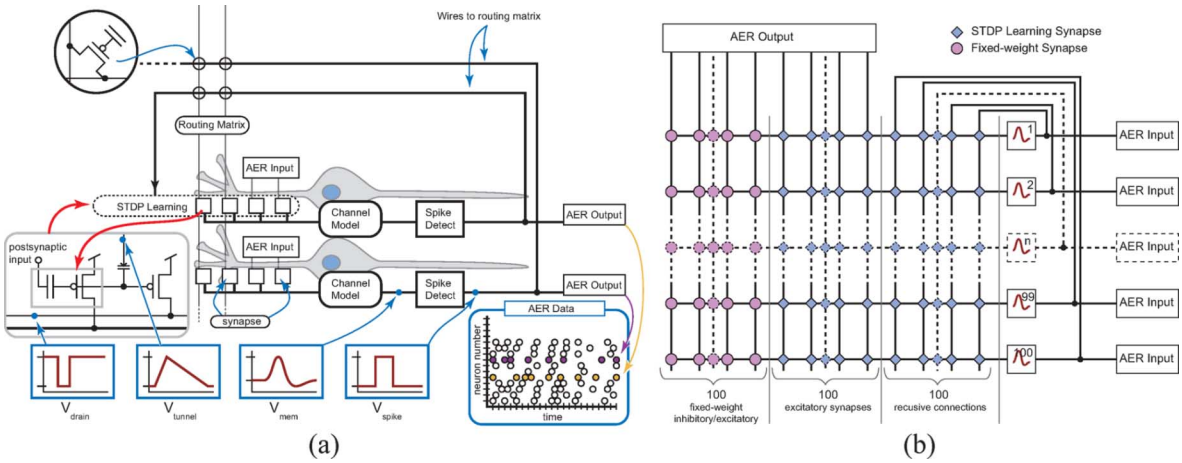

Fig. 2. The approach for the of biologically inspired neuron computation IC. (a) The system approach is to provide low-power biologically realistic silicon models of synapses (computation and learning), soma elements, and event (action potential) input and output interface handling through Address-Event Representation (AER). For this discussion, we treat the dendrites as a wire; treating the dendritic dynamics is beyond the scope of this discussion. (b) The architecture built of a densely packed, mesh-tye structure of synapses, an array of configurable elements to compile the desired soma dynamics, and AER blocks that output action potentials into the synapse fabric as well as AER blocks that that input action potentials from soma blocks. The synapse array is made up of three different functional blocks of synapses: 10,000 STDP excitatory learning enabled synapses connected in recurrent connections, 10,000 STDP excitatory learning enabled synapses connected from AER, and 10,000 programmable (fixed-weight) excitatory or inhibitory synapses connected from AER. The STDP synapses have regions of operation where we can turn off learning.

The following sections will the new Single IC Neuron Learning Array. Section II discusses Neuron IC architecture and infrastructure, including IC chip, configuration tools, and testing platform. Section III discusses components of Neuron IC, including soma circuits, non-adapting and adapting behavior of synapse circuits, and Address-Event Representation (AER) spike communication circuits. Section IV discusses measurement of Small Network of Neurons Section V discusses neuron Learning through STDP rules Section VI discusses neuron IC implementing a Ring WTA Topology.

II. Neuron IC Architecture and Infrastructure

This IC effort required the development of a unique IC architecture for coupling dense mesh arrays of synapses and groups of soma elements, as well as software tools for users to configure the chip as a network of neurons, as well as the resulting hardware testing infrastructure for the tools. We will discuss these three elements in turn over the following three subsections.

A. IC Chip Architecture

We first will present an overview and our design approach while developing the neural IC architecture.

Fig. 2(a) shows the high-level viewpoint of our IC implementation. We want to implement biological model circuits for synapses, soma (channel model), and input and output spikes efficiently into the system. We see the array of neurons as a specialized configurable array, similar to the development of Large-Scale Field Programmable Analog Arrays (FPAA) [10], [11], [14]. The reconfigurable nature of the platforms allow rapid building and testing of different circuit configurations. Because the typical spike rate for all of the neurons (100×200 events/s $= 20,000$ events/s) is much less than the speed of communication on the digital bus (> 1 M event/s), we can faithfully represent each event by directly transmitting or receiving the address of the event on that digital bus. Such communication approach is called Address-Event Representation (AER) [4]. Standard Control blocks for the programmable floating-gate circuit enables the FPAAs to provide area-efficient, accurately programmable analog circuitry [15]. Dendrites are modeled and implemented as loss-free wires.

This work shows the first integrated IC with network of neurons based on the original Single Transistor Learning Synapses (STLS) concept [6]. The STLS are modified EEPROM devices, fabricated in a standard CMOS process, that simultaneously provide long-term storage (non-volatile), computation, and adaptation in a single device. To get the highest achievable density for the array of synapses, we chose a mesh architecture for the interconnection fabric, as shown in Fig. 2(b), which results in general connectivity for this architecture. Our resulting architecture enables 100×100 excitatory recurrent learning synapses, that is connections from output of on-chip neurons to on-chip synapses, 100×100 excitatory learning synapses with external input from AER, and 100×100 inhibitory or excitatory fixed-weight synapses from external input from AER. In future versions, one could make the excitatory learning synapses be either excitatory or inhibitory synapses as was done for the programmable synapses; for this implementation, we only used excitatory learning synapses.

This approach shows the most direct approach of computing through a memory device, since in fact, the array of synapses is quite similar to an array of EEPROM devices, but with enhanced capabilities described above. Any off-chip memory based scheme will only be more complex and expensive system design. More configurable, sparse type patterns could be implemented in FPAA array type concepts. The analog programmability empowers the switch elements to have a dual role as computational elements [10], [13]. Because STLS synapses are at the density of EEPROM arrays, we expect similar STLS densities, particularly as we scale to smaller IC processes. Current EEPROM devices in 32 nm CMOS processes are less

TABLE I
PARAMETERS FOR THE NEURON IC CHIP

Parameter	Value
Number of Neurons	100
Prog Synapses	10000
STDP Synapses (Feedforward)	10000
STDP Synapses (Recurrent)	10000
STDP Prog	4.5V
Prog (inj)	5.6V
Erase (tun)	10.2V

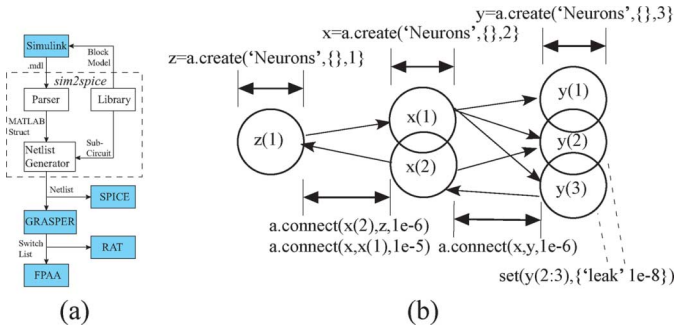

Fig. 3. Overview of our configurable toolflow. (a) Our traditional software flow for designing systems on the FPAA. Top level designs are done in Simulink. *Sim2Spice* converts it to a Spice netlist, which can then be compiled into an FPAA switch list. *GRASPER* does a place and route compilation from an annotated SPICE net list to a switch list for compiling on an FPAA device. *RAT* is a switch list visualization tool for the particular configurable device, allowing the user to move switch connections where desired. (b) Representation of our PyNN language for representing a basic network of neurons. This design flow can be compiled to object code that can configure our described IC for an appropriate neuron architecture. The current tool flow enables a PyNN text script to be compiled directly to a switch list in a parallel path to the SPICE compilation. A graphical representation built in Simuilnk can be compiled to PyNN code through Sim2spice.

than 100×100 nm^2 in area, as small or smaller than current integrated and functioning nano device array technologies.

Details of the soma elements, synapses, and AER communication system are presented in the following section. Some of the IC properties are summarized in Table I. We utilize industrial methods and practice of designing custom mask sets, allowing for 1000's of chips from a single batch of wafers. These aspects are essential when scaling these architectures from a single chip to a board (or multiboard) system. Further, these approaches show the technical viability for a commercial approach if/when an application is shown to be effective using neural ICs.

B. Tool Infrastructure for Configuring the Neuron IC

A key component of any configurable system is the need for tools to abstract the programming and configuration of the resulting system. Our previous configurable systems, Large-Scale Field Programmable Analog Arrays (FPAA), are enabled by a tool suite controlled through MATLAB [16], using Simulink [17] at the high level that compiles to a SPICE deck, which in turn, can be compiled [18] to programmable object code for the FPAA device. Fig. 3(a) illustrates the approach. Higher-level tools also enables the use of these systems in educational experiences [19], which we envision is one early application of these developed chips.

The tool flow for the chip of neurons is currently a simpler flow that could be integrated with the Simulink toolflow for building configurable systems. The base language we used for this approach is PyNN [20], rather than a SPICE deck, to specify the netlist level of the neuron structure; the PyNN approach is a network description of the resulting network. PyNN is designed to be a simulator-independent, Python-based open source language designed for describing spiking neural network models. In our case, we use this language definition for our tool compilation in Matlab, and the resulting compiled output can still be used in a PyNN platform. Fig. 3(b) shows our resulting PyNN description for a small neuron network. The resulting code in sequence is given below.

```
a=pynnSession                % create pynn object
x=a.create('Neurons',{},2) % create neurons
y=a.create('Neurons',{},3)

% change parameters
z=a.create('Neurons',{},1)  set(y(2:3),{'leak' 1e-8})

% define connections
a.connect(x,y,1e-6)
a.connect(x(3),z,1e-6)a.connect(z,x(1),1e-5)

a.compile                  % Compile (to switch list)

programNeuron(a.list)       % program floating gates

AER_run(stop_conditions)   % run experiment
```

As a result, PyNN language is somewhat higher level representation of the neural structure than a typical SPICE deck, enabling users to directly implement a network of neurons directly through this language. PyNN supports various software simulators and some neuromorphic hardware systems; one can port designs between these packages without modifying the experiment description.

Since these are configurable chips, we utilized our generic configurable IC platform for testing [16]. Platform was not optimized for these types of tests, so many measurements are noisier and lower resolution than possible. The focus of this paper is to show the functionality of this IC, rather than quantitative measurements.

III. COMPONENTS OF NEURON IC

This section is a systematic walk through the details of the soma elements, synapses, and AER communication system as well as showing corresponding measurement data. The key to building this bridge is utilizing all of the Si physics to model key biological physics, starting at the level of channels. For this implementation, a single neuron has 300 synapses feeding into the soma compartment, which is approaching a typical cortical neuron having 1000 to 10000 synapses. Modeling dendritic compartments would be the next significant milestone to achieving realistic neuron behavior; dendritic IC modeling are beyond the scope of this discussion. In the following sections, we will utilize these effects when discussing the network of neurons implemented on this IC.

We obtain biologically realistic dynamics by using established transistor channel models, published elsewhere, for models of passive and active channels, including bandpass (i.e., classic Na^+) and lowpass (i.e., classic K^+) channels [9], which build the foundation for our soma models, and that can be extended to dendritic models [2] We obtain biologically realistic dynamics by using established transistor channel models, published elsewhere, for models for non-adapting [7] and adapting [8] excitatory and inhibitory synapses, which build the foundation for our synapse models. Such dynamics have been analyzed [12] as well as models built from these components [11]. The remaining infrastructure simply provides communication between outputs of somas to the input of synapses. We will overview these models in the following subsections; more detailed treatment, including their biological relevance is discussed elsewhere [2], [7]–[9], [11], [12].

A. Soma Circuits

Fig. 4 shows the basic circuit diagram for the configurable soma block. From an FPAA perspective, these are the Computational Analog Blocks (CAB) for this infrastructure, with a unique routing infrastructure, that includes the programmed and adaptive synapses. Within this structure, we have the capability of multiple channels, infrastructure for communicating action potential events to other structures, as well as circuits to build local WTA feedback between the soma membrane voltages. We chose local WTA computation between soma elements to both emulate the similar effect in biological systems [21] as well as reduce the resulting neuron output event rate, which can be the primary source of power consumption if not properly handled. We have additional configurable infrastructure for allowing some level of debugging and tuning, for example, we have circuitry to directly perform a voltage clamp measurement on each channel element.

The base components are based on transistor channel models of biological channel populations [9]; we will summarize briefly the key concepts here. The physical principles governing ion flow in biological neurons share interesting similarities to electron flow through MOSFET channels, and exploiting these similarities results in dense circuits that model effectively biological soma behavior. The energy band diagram (source to drain) looking through the channel of the MOSFET is similar to the energy band diagram (inside to outside) looking through a biological channel. Because we utilize the similarities between biological and silicon channels, the voltage difference between the channel resting potentials on the silicon implementation is similar to the biological power supplies. The resulting spiking action-potential circuit requires six transistors, which is the same number of transistors and just a few more capacitors (transistor size capacitors) than the basic integrate and fire neuron approach [22].

In this IC, we have available one bandpass and one lowpass voltage-gated channel. We call the Na^+ channel a bandpass voltage-gated channel, meaning it has both an activating and inactivating mechanism causing current magnitude to increase and then decrease as time progresses. This channel circuit could be used to model a range of bandpass channel dynamics, within a range of parameters, and this channel has two time constants

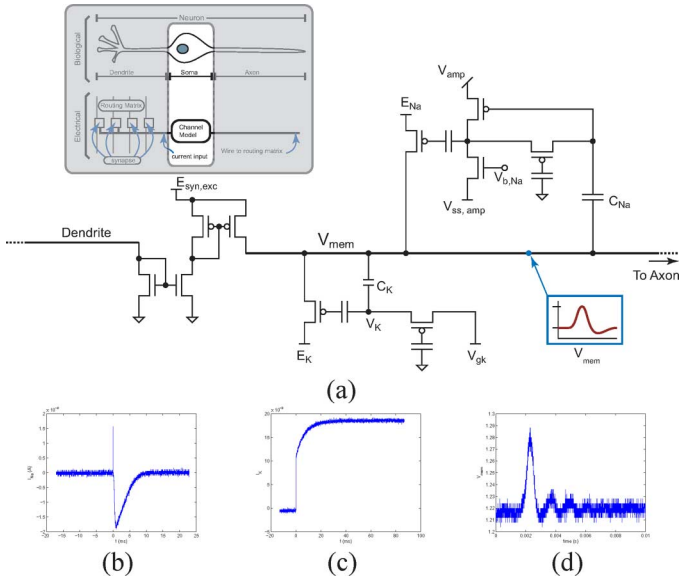

Fig. 4. Block diagram of the reconfigurable soma architecture. (a) The chip employs 100 of these configurable blocks. All blocks connect to the membrane voltage, and therefore the routing is straightforward; the output of the neuron goes through a single processing stage to communicate through the fabric. The small configurable block includes a low-pass channel, typical of a K channel, a bandpass channel, typical of a Na channel, leak channel, and nearest neighbor coupling to enable Winner-Take-All (WTA) behavior between the blocks. We level shift the resulting voltage for these blocks, which run at 2.5 V, as opposed to the adaptive synapse blocks running at a higher voltage to enable STDP. The output goes through a buffering circuit to transmit the signal either to the AER sender and/or back through the resulting synaptic fabric. (b) Voltage clamp results from a single neuron CAB element for Na channel. (c) voltage clamp results from a single neuron CAB element for K channel. (d) Representative trace of spontaneous spike waveform from combining the K and Na channel to make a neuron circuit.

(fast and slow). We call the K^+ channel a lowpass voltage-gated channel, meaning it has only an activating mechanism. This classic circuit implementation [9] leverages a large capacitive coupling from membrane to K gating voltage to achieve biological dynamics. Typically, the time constant is much slower than either of the time constants found in Na^+, resulting in a hyperpolarization of the cell for a period of time by the overshooting the equilibrium voltage during an action potential event.

Fig. 4 shows the voltage clamp responses for the Na and K channel models, as well as one of multiple representative output spikes. As seen in Fig. 4(b), a Na channel and gating function gives the step response of a bandpass filter, and in Fig. 4(c), a K channel and gating function gives the step response of a lowpass filter. Fig. 4(d) shows a resulting action potential from one of these pairs of channels in a soma block. These circuits are modified versions of the original circuit by adding additional floating-gate elements into the circuit to directly set the programmed bias currents (i.e., for timeconstants). The particular implementation reasonably implements the original circuit approach [9] using the floating-gate devices, but deviated from the original implementation in two ways. First, the gain of the Na^+ channel should have been higher, being closer to 3 to 4 instead of 8 (or higher) mentioned in [9], resulting in a less sharp transition for the resulting positive feedback, and making spiking behavior more challenging. Second, the Capacitive coupling into the K^+ channel should have been higher (more than a factor

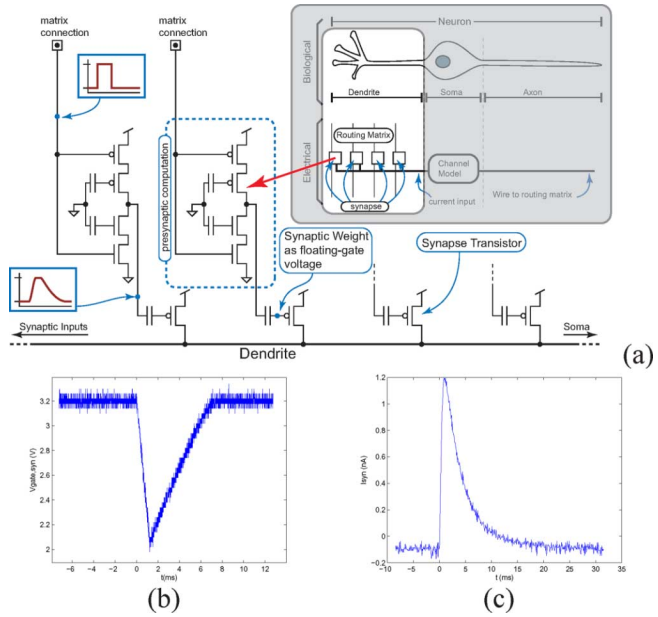

Fig. 5. Silicon Synapse Array utilizing a single floating-gate device for the synaptic array, giving densities similar to conventional EEPROM arrays. (a) Architecture for the synaptic array structure. The inputs into the array are preconditioned using a triangle waveform, such that, when the signal is passed through a subthreshold transistor, gives a biologically realistic post-synaptic potential. (b) A trace of the output of the gate waveform shaping circuit. (c) Measured synapse circuit current output during non-adapting behavior.

of 2), resulting in the channel enabling a significant restoring force instantaneously, and therefore making a spike more challenging. The coupling of these two nonidealities both makes tuning neuron dynamics more difficulty; overall dynamics were still able to be preserved, although the spike shape is not quite ideal as desired. Having a programmable and configurable infrastructure made using this chip design possible.

B. Synapse Circuits: Non-Adapting Behavior

Synapses represent the connection between axon signals and the resulting dendrite of a particular neuron. The connection starts as an electrical event arriving into the presynaptic cell, releasing chemicals that reach and modulate the channels at the postsynaptic cell, resulting in a response in the dendritic structure. A synaptic Post-Synaptic Potential (PSP) is modelled typically as

$$I_{syn} = te^{-t/\tau_{fall}} \tag{1}$$

where τ_{fall} is typically on the order of 0.5 ms to 2 ms.

Our synapse implementation, shown in Fig. 5(a), has a triangle waveform modeling the presynaptic computation, a MOSFET transistor modeling the postsynaptic channel behavior, and a floating-gate to model the strength of the resulting connection. We use a floating-gate device that can be used to store a weight in a nonvolatile manner, compute a biological EPSP, and demonstrate biological learning rules (discussed in the next subsection) [6]–[8]. A MOSFET transistor in subthreshold has an exponential relationship between gate voltage and channel current; therefore to get the resulting gate voltage to get the desired synapse current, we take a log of (1) to get the gate voltage, which has the shape of a triangle waveform. A typical measured triangle waveform from this chip is shown in Fig. 5(b), and the resulting PSP current is shown in Fig. 5(c).

Also seen in Fig. 5(a), as well as other earlier architecture figures, is that the architecture is highly scalable along a row of the computation, as well as scalable into a rectangular mesh architecture. For a mesh architecture, the presynaptic computation circuitry is placed at the top of the array and fed into all synapses afferent to the neuron in that column. The output from all synapses in one row feed into a soma block. Effectively, we have a modified EEPROM array with the computations required for a mesh-type synaptic array, resulting in the ultimate computing in memory architecture. The effective density for this 350 nm CMOS process is roughly 10,000 synapses per mm^2. This mesh architecture of synapses that supports all-to-all connectivity between soma blocks, effectively being a specialized routing fabric heavily involved in the computation.

Currently there is a significant debate about what the resulting precision for synapses would be; at this stage, there are many theories and approaches, but there is little agreement on those numbers. More research in neuroscience is essential in applying neuron inspired approaches as competitive applications before this question can be answered with high certainty. As a result, if one can easily and directly build synapses that can handle wide dynamic range and precision, it makes sense to include these possibilities. The synapse approaches we have built clearly enable both high dynamic range and precision in stored weight and in resulting dynamics. On the other hand, since our synapse element is one transistor (effectively a EEPROM), even if one wanted a single bit weight, the synapse circuit is still as simple as possible; a lower bit precision will simplify the programming infrastructure and learning circuits, but the core synapse transistor will be the same size. Any comparison of size of this synapse ($\approx$ one transistor) with other approaches must include the memory elements and communication to reach that element.

C. Synapse Circuits: Adapting Behavior

Biological synapses adapt to their environment of event inputs and outputs, where typical programming rules include long-term potentiation (LTP), long-term depression (LTD), and spike-time-dependent plasticity (STDP). In biology, synapses strengthen through chemical and morphological changes that improve signal transduction from the presynaptic to the postsynaptic cell [23], [24].

Recently, we presented a single floating-gate device that enabled both the long-term storage and PSP generation, but also allowed a family of LTP, LTD, and STDP type learning approaches through the same device [8]. In this neuron chip, we have implemented these learning algorithms as part of the array, and we will summarize the key aspects of the STDP learning algorithm. The weight increases when the postsynaptic spikes follow the presynaptic spikes and decreases when the order is reversed.

The learning circuitry is again placed at the edges of the array at the end of the rows, included in the soma blocks, therefore not limiting the area of the synaptic matrix/interconnection fabric. Fig. 6 shows the soma component with the additional blocks required for STDP learning. The row of synapses are

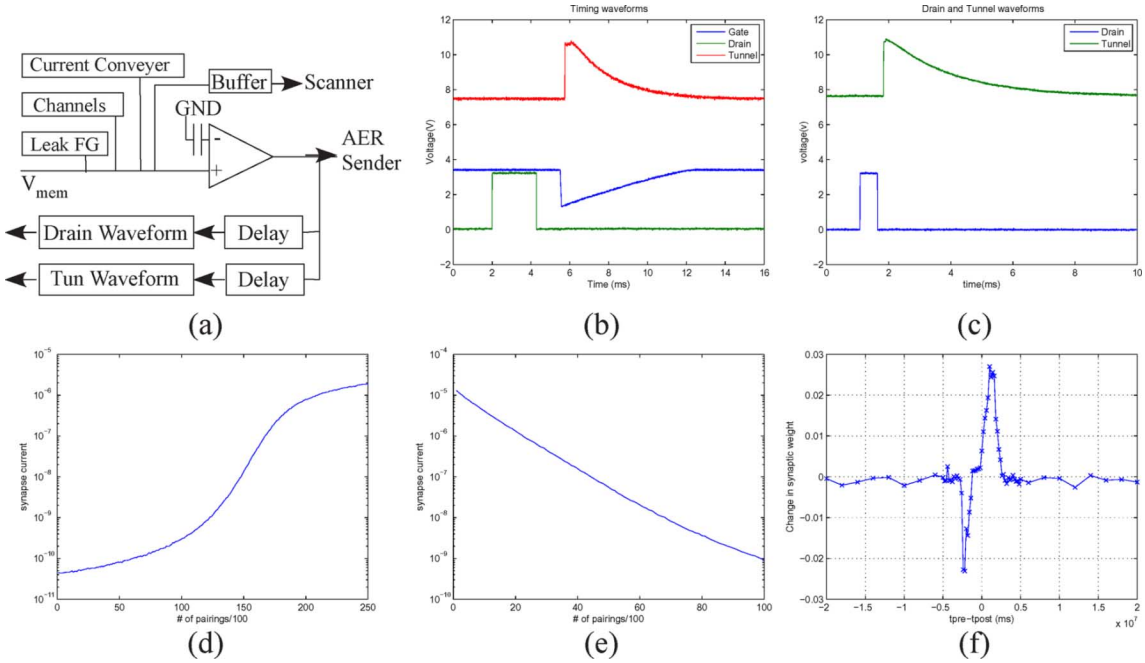

Fig. 6. STDP learning characterization from a single synapse element in the array. (a) Soma control circuitry including block diagram components needed to enable STDP operation. (b) Timing waveform (measured) occurring after an action potential event. (c) More timing waveforms. (d) Change in synapse strength for successive pre event first followed by post event with small delay. The experiment shows the effect of successive LTP (long-term potentiation) events. We plot measured synapse current, the measured value of the maximum synaptic output current, as a function of events. The effective synaptic weight is proportional to this current level. The effect shows the ability of both obtaining a large change in the dynamic range of the synaptic weight (over 4 orders of magnitude from below 100 pA to above 10 μA), as well as the capability to get a small percentage changes (16 percent worst case in these measurements) for the synaptic weight. (e) Change in synapse strength for successive post event first followed by pre event with small delay. The experiment shows the effect of successive LTD (long-term depression) events. (f) Typical measured STDP learning rule versus the delay between the presynaptic and postsynaptic spike.

turned into 'PROGRAM' mode for a short time, allowing the weight to adapt based on other signals in the inputs and then reset into 'RUN' mode. The learning mechanism is a combination of hot-electron injection (LTP) and Fowler–Nordheim tunneling (LTD). The learning mechanism is only enabled when an event on a postsynaptic neuron occurs. This approach allows a wide set of potential learning rules to be implemented by changing the resulting parameters. As part of the algorithm, we used programmable current-starved inverters to implement delays for the timing of the drain and tunneling pulses after the start of an output event. The presynaptic computation remains the same, effectively, from the PSP part of the computation, including a voltage triangle waveform at the input generated by the presynaptic computation block.

For STDP learning algorithm [8], at the occurrence of a postsynaptic spike, a program phase consisting of an injection pulse followed by a tunnel input is applied. If the input spike occurs before the output spike, the gate voltage is at a lower voltage, and enables more injection current on the floating-gate device, increasing the synapse weight. If the input spike occurs after the output spike, the gate voltage is at a higher voltage, and enables more tunneling current on the floating-gate device, decreasing the synapse weight.

Adding the STDP functionality makes no additional area complexity to the synapse block; we do have some additional timing control circuit required at the soma element which requires a small fraction of the soma area. Our discussion in [8], gives a detailed model as well as derivation of the LTP, LTD, and STDP learning rule. For completeness, we will summarize the key modeling results of the STDP learning rule. The approach is based physically on the hot-electron injection and tunneling mechanisms, with timing and voltage level of parameters being able to be programmed/configured for a given algorithm. We give the modeling we programmed/configured for this particular implementation, although one has significantly more flexibility as we reported previously [8]. In [8], we formulated the derivation using a simplified model of hot-electron injection current (I_{inj0}, I_{s0}, α, and V_{inj} are device level parameters, α close to 1)

$$I_{inj} = I_{inj0}\left(\frac{I_d}{I_{s0}}\right)^{\alpha} e^{\Delta V_{ds}/V_{inj}}, \tag{2}$$

and a simplified model of electron tunneling current (I_{tun0}, V_{ox} are device level parameters)

$$I_{tun} = I_{tun0} e^{(V_{tun}-V_{fg})/V_{ox}}, \tag{3}$$

along with the synapse model in (1), and a controlled decay in the tunneling voltage from its maximum value (t/τ_{cp}) to formulate the model of the STDP circuit behavior. We get the following weight update (Δw) as a function of the weight value (w) as a function of the delay (t_d) between the presynaptic and postsynaptic event

$$\Delta w = A e^{\Delta V_{ds}/V_{inj}} w^{1+\alpha} T_{inj} e^{\alpha t_d/\tau_{fall}} - B T_{tun} w^{1+\beta} e^{t_d/\tau_{cp}} \tag{4}$$

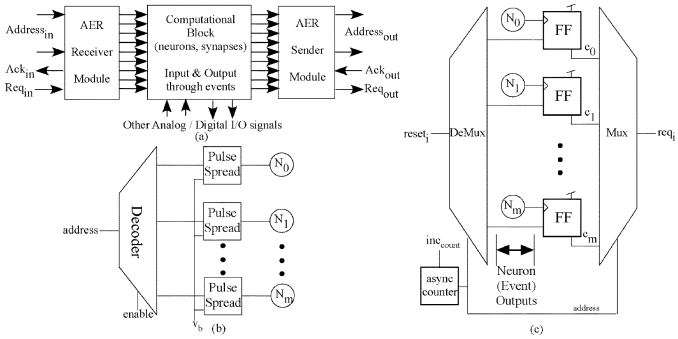

Fig. 7. Block diagram of our Address Event Representation (AER) communication scheme. (a) AER approach is used to interface the input and output events (action potentials) of a network of neurons to the outside world. We use an AER receiver to get input events, and an AER transmitter to send output events. (b) AER receiver block diagram. (c) AER transmission block diagram.

where T_{inj} and T_{tun} are the pulse widths of the injection and tunneling phases, respectively, and A, B, and β are combinations of device constants that can be modified by circuit parameters [8]. This single formulation showed a wide range of STDP curves that were possible from the circuit.

D. Spike Communication Through AER

The user of the neuron IC would need to communicate events on and off the IC, as seen in Fig. 7(a). We use AER for both the input and output events. Because the typical spike rate for all of the neurons ($100 \times 200\ \mathrm{events/s} = 20{,}000\ \mathrm{events/s}$) is much less than the speed of communication on the digital bus ($> 1\ \mathrm{M\,event/s}$), we can faithfully represent each event by directly transmitting or receiving the address of the event on that digital bus. Further, AER provides a standard for interfacing to these chips either with additional chips or with other chips using the AER standard.

Our one-dimensional, asynchronous AER systems were designed considering that we can consider a minimum time resolution for our events after the start of a single event, and therefore can gather/queue the resulting events accordingly. Our system was designed for rates above 10 M events/s, but none of the measured or proposed applications require such an event rate. Achieving this rate would require additional design for the resulting board infrastructure (i.e., USB 1.1 won't communicate at this speed off of the board). The queuing approach enables easy interfacing to synchronous systems, and we have written microcontroller code on our test system to enable AER interfacing. Our AER system conforms to the typical AER asynchronous definition for communication. Because we wanted our AER approach to scale to a wide range of CMOS technologies, we developed our approach as compilable code in Cadence toolset; we believe this is the first case of a compiled AER system reported. Given the size of our networks, only one-dimensional sender and receiver modules were necessary and simplify the resulting AER implementation. Details of the AER system are beyond the scope of this paper, and will be presented elsewhere.

The AER output module [Fig. 7(b)], which is taking output neuron events and transmitting an address for that particular event, takes 100 rising-edge triggered inputs and latches all of the events in a given time window into an array of N flipflops. The result is a simple arbitration circuit, and is easily scalable to using asynch FIFO queues on event stations. The latched structure of stored events then converts each event, in turn, to an address on the output bus. We implemented multiple debugging options, some that are useful in interfacing the output event stream to digital microcontroller. The working system was completely synthesized from verilog to standard cells and placed and routed to silicon in roughly 100 lines of code, where 20 are port/wire declarations, and is parametrizeable.

The AER input module [Fig. 7(c)], which is taking input event addresses and communicating events to the synaptic array, outputs 200 events into the neural array. the functionality is primarily a digital decoder followed by a circuit to hold the event for a particular duration to be input into the triangle waveform generation. Again, the entire structure was synthesized, placed and routed to silicon from verilog.

We measured input events to and output events from spiking neurons creating events that are communicated through the AER communication system. The system uses the input system to stimulate the system to create these events. The resulting system is fast enough to communicate input and output events for exhaustive testing this entire IC.

IV. Measurement of Small Network of Neurons

In the following subsections, we will present data measured from small networks of neurons compiled on this IC. First, we discuss the approach and impact by programming the floating-gate elements to eliminate mismatch, a critical issue for most neuromorphic IC implementations. Second, we will describe the basic interactions between two neurons when we compile two neurons together in a single device, as well as describe the results when we compile a group of neurons as a single chain of neurons with and without a feedback connection. Finally, we will describe experimental requires when we compile a group of neurons in a Winner-Take-All (WTA) configuration.

As part of the fabrication process, we obtained and tested multiple ICs; we have infrastructure to measure at least three ICs in parallel. Although we have not done detailed statistical testing over a complete lot of chips, the variability between chips after programming target currents for the synapses and somas was not noticed over 3 months of testing 8–10 ICs. We have taken the same or similar data over multiple ICs and not seen any noticeable differences. Being able to directly program bias currents, not to mention more direct calibration of threshold voltage levels (as mentioned in the first subsection), allows for highly repeatable results from a densely packed analog computing system.

A. Eliminating Mismatch By Programming Floating-Gate Elements

Transistor mismatch has been a classical issue for neuromorphic designs [22], as well as all designs. Early neuromorphic systems had to deal with mismatch issues (i.e., [25]) to get even first order performance for their particular ICs. Although many have approached the problem by stating there is mismatch in neural systems, and therefore we should just use this mismatch,

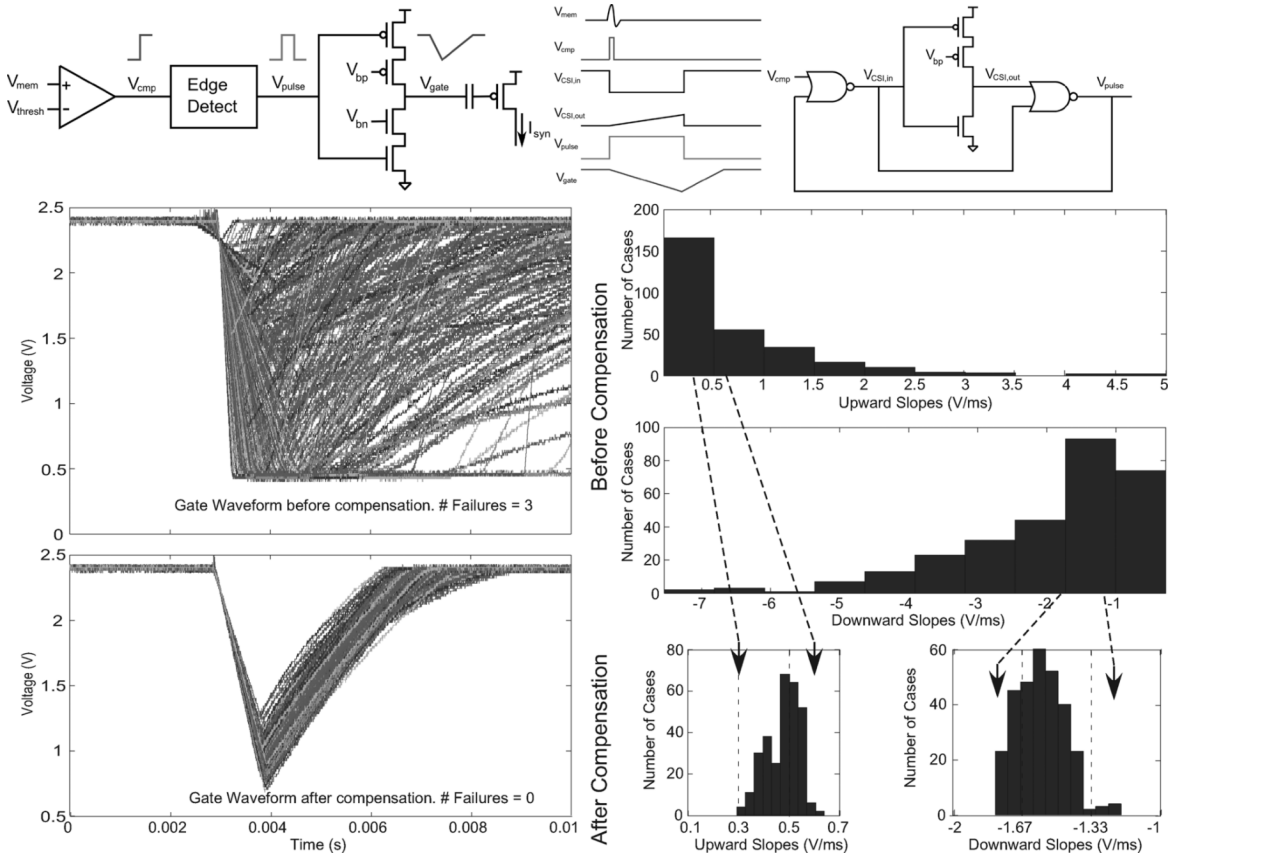

Fig. 8. An example of solving mismatch problems for one of our implemented neural structures. For our synapse structures, we can have threshold voltage mismatch both in the edge detection/pulse generation circuits, ramp generator, as well as synapse channel transistor (actual circuit has additional circuit complexity, but we keep things clear for this discussion) In each of these cases we have floating-gate devices to tune the particular parameters, and in particular, directly affect the issue with threshold voltage mismatch. We show the initial curves after initial programming, in which the reference devices were all set to the same current, illustrating the direct issue of these multiple threshold voltage issues; this plot is very characteristic of neuromorphic systems built over the last two decades. We also show the result after calibrating just the threshold voltage mismatch for the ramp generator, resulting in a change to very usable characteristics. More tuning can make the curves further ideal going forward.

to date, no system has effectively utilized these errors, and usually systems simply require more resources (e.g., grouping several neurons) to make such a system functional. A more realistic approach seems to be to enable a system that can deal with the mismatch and then allow the level of mismatch that might be useful for the particular system of interest.

The primary source of mismatch in subthreshold or near threshold circuits is threshold voltage mismatch. For a typical MOSFET device, the average threshold voltage mismatch(ΔV_{T0}) is typically 10 mV or larger; since the effect of mismatch on the transistor current is proportional to $\exp(\kappa \Delta V_{T0}/U_T)$, a 10 mV offset results in roughly 20% mismatch between devices. Minimum size devices will be significantly higher mismatch effects. One can reduce mismatch effects by using larger transistors, since the average mismatch will decrease roughly as the square root of the device size, resulting in far lower system density as well as lower power efficiency due to the larger capacitances.

The use of floating-gate devices enables programmable elements that could be used to remove mismatch effects. Floating-gate techniques directly compensate for threshold-voltage mismatch effects by directly supplying an offset charge that can compensate for the oxide or interface charge responsible for ΔV_{T0}. Fig. 9 shows experimental data from our chip illustrating the process and impact of correcting for mismatch that we use throughout our floating-gate chip. The data for the triangle waveforms connecting to the gate of our post-synaptic transistor model before correction looks nearly unusable, but is the typical situation for most neuromorphic chips (i.e., [25]). When looking at the distribution of the errors, we find a change by as much as a factor of three or more from the baseline level, consistent with the threshold voltage mismatch from multiple devices. The plot after correcting the errors in an automated fashion (not hand tuned for optimal performance) of just the gate waveform generator circuit's three floating-gate elements dramatically improves the characteristics, as seen by the improvement of the resulting upgoing and downgoing slope.

The approach for the correction required initially programming the devices to a ideal (and somewhat lower) current level, measuring the resulting triangle wave devices, extracting the resulting down going and up going slopes for each device, extract the resulting mismatch in the bias current, and reprogram using hot-electron injection according to this mismatch profile (and repeat if necessary) Because our corrections are by injection, we typically program the devices to slightly lower currents

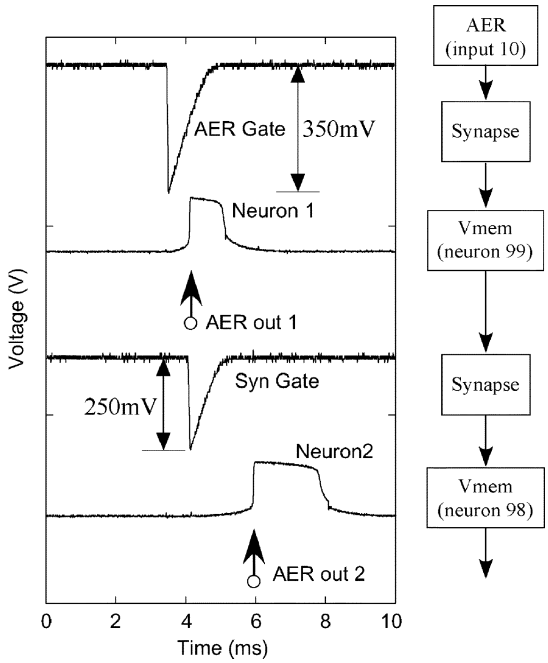

Fig. 9. Illustration showing the connection, signals, and approach for connecting two neurons together originating from an external source. This setup starts with an AER input (input 10) from the programmable synapses that is configured to be an excitable synapse. The input creates a response in the membrane of neuron 99, which is programmed to be strong enough to cause that neuron to create an action potential. The action potential is recorded as an event through the AER communication system. That output is also connected to the synapse on the membrane of neuron 98. The synapse is again programmed strong enough to create an action potential, which is also recorded through the AER communication system.

so we can make corrections entirely by injection programming and not requiring an erase of the parameters; the percent current mismatch will be roughly the same when programming for a slightly lower current level. This synapse calibration function requires 3–5 minutes, and is mostly limited to the MATLAB level interface and communication; a dedicated piece of micro controller code would take significantly less time. In some cases, the mismatch was so significant, that we did not get a proper triangle wave curve; depending if one or both slopes could not be extracted, we moved the parameters to get some triangle wave performance and then iterate as mentioned above.

Improving other parameters in this approach will further bring the elements to an ideal state; we use these techniques throughout our IC to get sufficient performance. This approach is different from other approaches (i.e., [26]) in that we are directly correcting the mismatch of the devices at a device level in what is a straightforward automated approach; techniques that tune at neuron, connectivity, or system level can still be employed to further improve performance, starting at a much closer optimum point.

B. Neuron IC Implementating a Chain of Neurons

Once the system mismatch is sufficiently handled for our components, the next stage is illustrating the connection configuration issues between two neurons through the resulting chip infrastructure. Fig. 9 shows a diagram and experimental data of

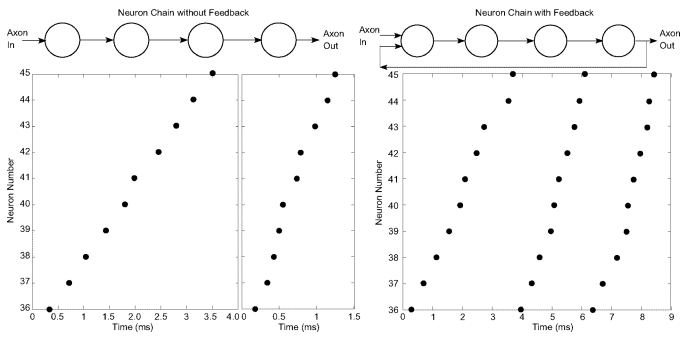

Fig. 10. Block diagram for a chain of neurons, including a chain of neurons without a feedback connection. and a chain of neurons with a feedback connection and additional excitatory input. The black dots indicate an event recorded from a particular neuron at that particular point in time. For the case with no feedback, we measured these structures with two different sets of synaptic strengths, resulting in different delays between the neurons.

two neurons compiled on this IC. This characterization can be repeated, pairwise for any two neuron elements.

Fig. 10 shows two examples of a chain of neurons, both with and without feedback. These neurons were programmed with some mismatch compensation, although the channel models were not programmed using mismatch compensation.

C. Neuron IC Implementing a Ring WTA Topology

To show computation between a group of neurons, we implemented a ring Winer-Take-All (WTA) topology of neurons, as shown in Fig. 11. We chose this model since it is one of the few neuron models where the resulting network computation is understood [21]. The network is composed of multiple (N) excitatory neurons that all synapse (excitatory synapses) onto a single neuron that provides inhibitory feedback connection to all of the original neuron elements, resulting in a WTA type behavior as discussed previously [27]. Further, by having local reciprocal inhibitory connections, one can make the WTA network a locally winning network, similar to WTA networks with horizontal diffusor connections between neighbor neurons.

Fig. 11 also shows experimental data from this group of neurons. The strength of the excitatory connection between each excitatory neuron and the center inhibitory neuron is programmed identical for all excitatory neurons. The strength of the inhibitory connection between the center inhibitory neuron and each excitatory neuron are programmed to identical values, although the strength and time duration of the inhibition time was stronger than the excitatory inputs. We show two cases for different relative synaptic strengths. The synaptic strength from each AER input signal to each excitatory ring neuron is different. We applied device-level techniques to eliminate the mismatch at the synapses and neurons, as described above. We apply random input events into each neuron through a single synapse (we record the input events), and record the resulting measured output events. We get expected resulting behavior, both that we get a repeated firing from the inhibitory neuron (46) that regulates and decreases the firing of the other excitatory neurons, and that when locally in time when we get a strong input burst of activity at a particular neuron, we get a high probability of a resulting output event at that neuron.

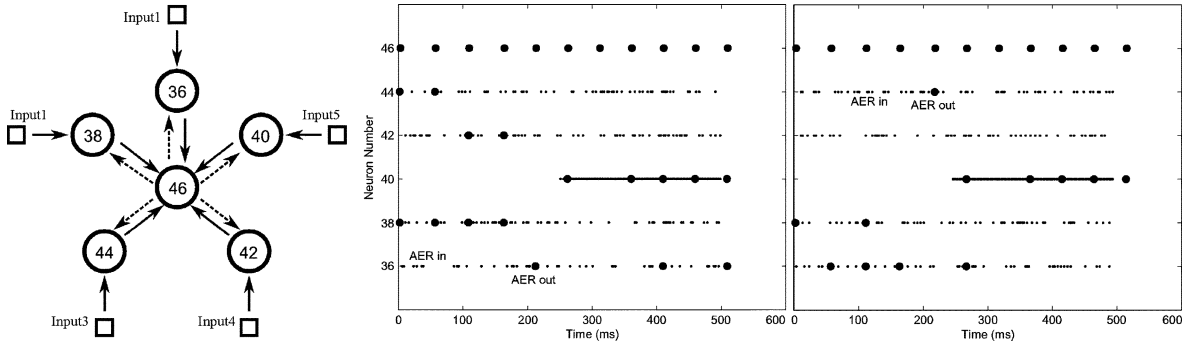

Fig. 11. Diagram of a spiking Winner-Take-All (WTA) network drawn as a ring topology. The network is a combination of input neurons on the outside part of the ring, and an interneuron, which provides the negative feedback through inhibitory neurons, for the WTA computation. We label the particular neurons used for this experiment, and all synapses used are programmable synapses, whether excitatory or inhibitory. We show experimental data from a small network from the IC, where the small, lighter dots are the input events into the structure, and the larger black dots are the output events from the structure. The interaction with the inhibitory neuron, and resulting inhibitory synapses greatly decreases and refines the resulting firing rate at the output. We have excitatory inputs into each of the neurons around the outer ring for this WTA network. We provide two data graphs for the same input stream but with different inhibitory synapse strengths. To enable local winning networks, we also have additional inhibitory connections between neighboring neurons.

TABLE II
COMPARISON OF SYNAPSE DENSITY AND FUNCTION OF WORKING IMPLEMENTATIONS

Chip Built	Process Node (nm)	Die Area (mm^2)	No of Synapses	Synapse Area (μm^2)	Syn density	Synapse Storage Resolution and Complexity
GT Neuron1d	350	25	30000	133	**1088**	$>$ 10bit, STDP
FACETs IC [29], [30]	180	25	98304	108	3338	4bit register, STDP
Stanford STDP	250	10.2	21504	238	3810	STDP, no storage
INI IC 1 [31]	800	1.6	256	4495	7023	1bit w/ learning dynam
INI Chip 2 [32]	350	68.9	16384	3200	26122	2.5 w/ learning dynam

D. Modeling Energy Efficiency of the Neuron Array

In this subsection, we discuss the power consumed by the network of neurons . For these discussions, we will assume an average neuron event rate of 10 events/s, $\mathrm{V}_{dd} = 2.4\ \mathrm{V}$, a long line of shared synapse inputs/outputs on die is 1 pF, $\tau_{fall} = 1\ \mathrm{ms}$, and external, off-chip capacitances of 10 pF. The synapses require a burst (PSP) of current for each event, where the average time for the output is 2 τ_{fall}, at a peak current determined by driving the line to the soma (i.e., 2 nA). The resulting energy for the synapse output per spike would be roughly 10 pJ/spike, and roughly the same energy at the input of communication/generation of the triangle (pre-synaptic) event. The spike generation aspect of the soma requires significant Na^{+} current to create sufficient positive feedback. The total number of synaptic inputs gives a measure of the current level required to reach a threshold; a typical number would be the maximum synaptic current times the square root of the number of inputs. For the case of 300 inputs, we would expect 170 pJ/output event. The AER communication interface needs to handle the resulting input and output events, involving charging capacitances (static power is ≈ 0 to set up the single event input line as well as pack up and transmit resulting events. Average numbers show 35 pJ/input event and 240 pJ/output event; the output events are higher because of driving external communication lines.

For an entire chip of 100 neurons, 200 inputs, and 30,000 synapses all operating with average 10 events/second, we get synapse power of 6 μW, Neuron Power of 0.2 μW, and AER communication Power of 0.3 μW. The synapse power consumes most of the resulting power. For a small network (6 neurons, 2 synapses per neuron), we are looking at a power consumption of 30 nW. This power level is negligible compared to the protection circuitry used around the IC ($\approx$ 10 $\mu\mathrm{A}$), as well as the resulting programming circuitry; although we powered the programming circuitry during programming, it could be disabled when operating the network. These power numbers don't include buffering instrumentation circuitry which are unneeded (and are turned off) during chip operation.

We can compare this power level to a digital computation computed using say a typical 4th order RK method. We would expect roughly 200 function evaluation MACs per iteration per neuron, and roughly 20 function evaluation MACs per iteration per synapse. One would expect a fixed sample rate of 20 kSPS would be required; for example 20 kSPS is the low-end sample rate used for dynamic clamp experiments. Therefore for a full system of 100 neurons and 30000 synapses, we would require roughly 12.5 GMACs each second. Given the best case for digital computation, 32-bit MAC operation(required for accurate ODE solutions) at the best case power efficiency of 100 pJ [28] and ignoring communication power (which is usually larger), the system would require 1.25 W compared to the $<$ 7 μW required for our IC.

We still have quite a long way to go before reaching the 10 pW/neuron power efficiency we see in cortical cells in the human brain. For our system, a single neuron with synapses requires 63 nW of power. Cortical cells, which also have extensive dendritic compartments that are available for computation, are a factor of 6000 from our efficient implementation. Decreasing the average spike rate to biological levels of one event per 2 seconds, and decreasing the power supply from 2.4 V to 150 mV supply will improve this factor by 300. The remaining factor of 20 could be achieved when scaling IC processes, decreasing the source-drain to well/substrate capacitances, and proportionally increasing the power efficiency. Such a chip built

in a 45 nm technology should have similar power efficiency as biological neurons, but a real comparison would require implementing dendritic approaches as well as increasing the number of synaptic inputs to biological levels of 1000 to 10000.

V. Conclusion

We presented a single-chip array of 100 biologically-based electronic neuron models interconnected to each other and the outside environment through 30,000 synapses. The chip was fabricated in a standard 350 nm CMOS IC process. Our approach used dense circuit models of synaptic behavior, including biological computation and learning, as well as transistor channel models. We used Address-Event Representation (AER) spike communication for inputs and outputs to this IC. We presented the IC architecture and infrastructure, including IC chip, configuration tools, and testing platform. We presented measurement of small network of neurons, measurement of STDP neuron dynamics, and measurement from a compiled spiking neuron WTA topology, all compiled into this IC.

Table II shows the structure presented in this paper results in the best synaptic density over other ICs built to date [29]–[32]. We define synapse density as the synapse area normalized by the square of the process node. Further, we achieve this synapse density in a working neural array with synapse complexity capable of high storage as well as STDP behavior; these techniques will scale down and have relatively similar density to EEPROM density at a given process node. These results demonstrate the resulting advantage of floating-gate approaches for neuromorphic engineering applications.

References

[1] E. Farquhar, D. Abramson, and P. Hasler, "A reconfigurable bidirectional active 2 dimensional dendrite model," in *Proc. Int. Symp. Circuits and Systems*, May 2004, vol. 1, pp. 313–316.

[2] S. Nease, S. George, P. Hasler, S. Koziol, and S. Brink, "Modeling and implementation of voltage-mode CMOS dendrites on a reconfigurable analog platform," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 1, pp. 76–84, Feb. 2012.

[3] S. Saighi, Y. Bornat, J. Tomas, and S. Renaud, "A library of analog operators based on the Hodgkin-Huxley formalism for the design of tunable, real-time, silicon neurons," *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 1, pp. 3–19, Feb. 2011.

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

[5] J. Schemmel, J. Fieres, and K. Meier, "Realizing biological spiking network models in a configurable wafer-scale hardware system," in *Proc. IEEE Int. Joint Conf. Neural Networks*, 2008, pp. 969–976.

[6] P. Hasler, C. Diorio, B. Minch, and C. Mead, "Single transistor learning synapse with long term storage," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 1995, vol. 3, pp. 1660–1663.

[7] C. Gordon, E. Farquhar, and P. Hasler, "A family of floating-gate adapting synapses based upon transistor channel models," in *Proc. Int. Symp Circuits and Systems*, May 2004, vol. 1, pp. 317–320.

[8] S. Ramakrishnan, P. E. Hasler, and C. Gordon, "Floating gate synapses with spike time dependent plasticity," *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 3, pp. 244–252, Jun. 2011.

[9] E. Farquhar and P. Hasler, "A bio-physically inspired silicon neuron," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 3, pp. 477–488, Mar. 2005.

[10] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. M. Twigg, and P. Hasler, "A floating-gate based field programmable analog array," *IEEE J. Solid State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sep. 2010.

[11] A. Basu, S. Ramakrishnan, C. Petre, S. Koziol, S. Brink, and P. E. Hasler, "Neural dynamics in reconfigurable silicon," *IEEE Trans. Biomed. Circuits Syst.*, vol. 4, no. 5, pp. 311–319, Oct. 2010.

[12] A. Basu and P. E. Hasler, "Nullcline based design of a silicon neuron," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 11, pp. 2938–2947, Nov. 2010.

[13] C. Twigg, J. Gray, and P. Hasler, "Programmable floating-gate FPAA switches are not dead weight," in *Proc. Int. Symp. Circuits Syst.*, May 2007, pp. 169–72.

[14] D. Abramson, C. Schottmann, and P. Hasler, "Programmable and configurable MITE systems enabled through precise floating-gate programming," *IEEE Trans. Circuits Syst. I, Reg. Papers*, submitted for publication.

[15] A. Basu and P. E. Hasler, "A fully integrated architecture for fast and accurate programming of floating gates over six decades of current," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 6, pp. 953–962, 2011.

[16] S. Koziol, C. Schlottmann, A. Basu, S. Brink, C. Petre, S. Ramakrishnan, and P. Hasler, "Hardware and software infrastructure for a family of floating-gate FPAAs," in *Proc. IEEE Int. Symp. Circuits and Systems* , 2010.

[17] C. R. Schlottmann, C. Petre, and P. E. Hasler, "Simulink framework for design to and automated conversion on large-scale FPAA devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, submitted for publication.

[18] F. Baskaya, D. V. Anderson, P. Hasler, and S. K. Lim, "A generic reconfigurable array specification and programming environment (GRASPER)," in *Proc. Eur. Conf. Circuit Theory and Design*, Aug. 2009, pp. 619–622.

[19] C. Twigg and P. Hasler, "Incorporating large-scale FPAAs into analog design and test courses," *IEEE Trans. Educ.*, vol. 51, no. 3, pp. 319–324, Aug. 2008.

[20] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. A. Pecevski, L. Perrinet, and P. Yger, "PyNN: A common interface for neuronal network simulators," *Front. Neuroinform*, 2008.

[21] G. Indiveri, T. Horiuchi, E. Niebur, and R. Douglas, "A competitive network of spiking VLSI neurons," in *Proc. World Congress Neuroinformatics*, Vienna, Austria, Sep. 2001.

[22] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[23] G.-Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength and post-synaptic cell type," *J. Neurosci.*, vol. 18, pp. 10464–10472, 1998.

[24] H. Markram, J. Lubke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps," *Science*, vol. 275, pp. 213–215, 1997.

[25] L. Watts, D. Kerns, R. Lyon, and C. Mead, "Improved implementation of the silicon cochlea," *IEEE J. Solid-State Circuits*, vol. 27, no. 5, 1992.

[26] E. Neftci, E. Chicca, G. Indiveri, and R. Douglas, "A systematic method for configuring VLSI networks of spiking neurons," *Neural Comput.*, vol. 23, no. 10, pp. 2457–2497, 2011.

[27] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of $\mathrm{O(N)}$ complexity," in *Advances in Neural Information Processing Systems 1*. San Mateo, CA: Morgan Kaufmann, 1989.

[28] H. B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Minimization of energy per Op in an asynchronous pipeline above and below threshold," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.

[29] J. Schemmel, A. Grübl, K. Meier, and E. Muller, "Implementing synaptic plasticity in a VLSI spiking neural network model," in *Proc. Int. Joint Conf. Neural Networks*, Vancouver, BC, Canada, 2006, pp. 1–6.

[30] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," in *Proc. IEEE Int. Joint Conf. Neural Networks*, Hong Kong, pp. 431–438.

[31] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Trans. Neural Networks*, vol. 17, no. 1, pp. 211–211, Jan. 2006.

[32] P. Camilleri, M. Giulioni, V. Dante, D. Badoni, G. Indiveri, B. Michaelis, J. Braun, and P. del Giudice, "A Neuromorphic aVLSI network chip with configurable plastic synapses," in *Proc. 7th Int. Conf. Hybrid Intelligent Systems*, 2007, pp. 296–301.