Synthesizing cognition in neuromorphic electronic systems

Emre Neftci[a,1], Jonathan Binas[a], Ueli Rutishauser[b], Elisabetta Chicca[a,c], Giacomo Indiveri[a], and Rodney J. Douglas[a]

[a]Institute of Neuroinformatics, University of Zurich and Eidgenössiche Technische Hochschule Zurich, 8057 Zurich, Switzerland; [b]Department of Neural Systems, Max Planck Institute for Brain Research, 33615 Frankfurt am Main, Germany; and [c]Department of Neural Systems, Cognitive Interaction Center of Excellence, University of Bielefeld, 60438 Bielefeld, Germany

Edited* by Terrence J. Sejnowski, Salk Institute for Biological Studies, La Jolla, CA, and approved June 10, 2013 (received for review July 20, 2012)

The quest to implement intelligent processing in electronic neuromorphic systems lacks methods for achieving reliable behavioral dynamics on substrates of inherently imprecise and noisy neurons. Here we report a solution to this problem that involves first mapping an unreliable hardware layer of spiking silicon neurons into an abstract computational layer composed of generic reliable subnetworks of model neurons and then composing the target behavioral dynamics as a "soft state machine" running on these reliable subnets. In the first step, the neural networks of the abstract layer are realized on the hardware substrate by mapping the neuron circuit bias voltages to the model parameters. This mapping is obtained by an automatic method in which the electronic circuit biases are calibrated against the model parameters by a series of population activity measurements. The abstract computational layer is formed by configuring neural networks as generic soft winner-take-all subnetworks that provide reliable processing by virtue of their active gain, signal restoration, and multistability. The necessary states and transitions of the desired high-level behavior are then easily embedded in the computational layer by introducing only sparse connections between some neurons of the various subnets. We demonstrate this synthesis method for a neuromorphic sensory agent that performs real-time context-dependent classification of motion patterns observed by a silicon retina.

decision making | sensorimotor | working memory | analog very large-scale integration | artificial neural systems

Unlike digital simulations, in which the dynamics of neuronal models are encoded and calculated on general purpose digital hardware, "neuromorphic" emulations express the dynamics of the neural systems directly on an analogous physical substrate (1). Digital simulations have the advantage that they can be exactly and reliably programmed using numerical operations of very high precision. However, they suffer the disadvantage that they are cast on abstract binary electronic circuits whose operation is entirely divorced from the physical processes being simulated. Consequently, such simulations do not readily advance our understanding of how biological neural systems are able to attain their extraordinary physical performance, using only large numbers of apparently unreliable, slow, and imprecise neural components, a problem first recognized by von Neumann more than a half century ago (2). Furthermore, the reliability of digital systems comes at high cost of the circuit complexity necessary for the orchestration of communication and processing, an overhead that declares itself also in the costs of system construction and power dissipation (3).

The alternative, neuromorphic, approach to information processing strives to capture in complementary metal-oxide semiconductor (CMOS) very large-scale integration (VLSI) electronic technology the more distributed, asynchronous, and limited precision nature of biological intelligent systems (1, 4).[†,‡] Research in this domain is now quickly accelerating (5–7), spurred on by radical changes in the design of computing architectures to favor highly distributed processing, as well as the economic promise of emulating the efficient intelligence of biological brains. So far, research emphasis has been on developing large-scale neural-like electronic systems (6–8). However, the concepts and methods for installing the dynamics necessary to express cognitive behaviors on these substrates are relatively poorly developed, mainly because the deep question of how biological brains install cognition on their neural networks remains open. It is to this behavioral configuration problem that our paper makes its contribution. We describe here an architecture and method for embedding simple cognitive behaviors in a real-time neuromorphic CMOS VLSI system. We refer to this behaving system as a neuromorphic "agent" because it is an autonomous entity that observes and acts on its environment through sensors and effectors, and its behavior is coherently directed toward achieving an abstract goal.

The hallmark of cognitive behavior is the ability of an agent to select an action based not only on specific external stimuli, but also on their context (9). Animals can learn such state-dependent sensorimotor mappings with formidable efficiency (10, 11). The likely reason for this ability is that evolution has learned how to construct modular brain circuits (12) that can be easily assembled to provide a useful spectrum of behaviors through learning on relatively few dimensions (13). Unfortunately, as engineers we do not have the benefit of rapid evolution in very large search spaces, and so we must instead combine insights from biology together with engineering principles to invent neuromorphic circuit modules that

Significance

Neuromorphic emulations express the dynamics of neural systems in analogous electronic circuits, offering a distributed, low-power technology for constructing intelligent systems. However, neuromorphic circuits are inherently imprecise and noisy, and there has been no systematic method for configuring reliable behavioral dynamics on these substrates. We describe such a method, which is able to install simple cognitive behavior on the neuromorphic substrate. Our approach casts light on the general question of how the neuronal circuits of the brain, and also future neuromorphic technologies, could implement cognitive behavior in a principled manner.

Author contributions: E.N., J.B., U.R., E.C., G.I., and R.J.D. designed research; E.N. and J.B. performed research; E.N. and J.B. contributed new reagents/analytic tools; E.N. and J.B. analyzed data; and E.N., J.B., U.R., E.C., G.I., and R.J.D. wrote the paper.

The authors declare no conflict of interest.

*This Direct Submission article had a prearranged editor.

Data deposition: The experimental data can be found under http://ncs.ethz.ch/projects/vlsi-wta-networks/synthesizing-cognition-in-neuromorphic-vlsi-systems-experimental-data/view. Simulations scripts can be found under http://ncs.ethz.ch/projects/vlsi-wta-networks/synthesizing-cognition-in-neuromorphic-vlsi-systems-scripts/view. This website is provided by Eidgenössiche Technische Hochschule (ETH) Zurich and maintained by the Neuromorphic Cognitive Systems (NCS) group at the institute of Neuroinformatics, University of Zurich and ETH Zurich.

[1]To whom correspondence should be addressed. E-mail: emre@ini.phys.ethz.ch.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1212083110/-/DCSupplemental.

[†]Telluride neuromorphic cognition engineering workshop. http://ine-web.org/workshops/workshops-overview, 1994–2012.

[‡]The Capo Caccia workshops toward cognitive neuromorphic engineering. https://capocaccia.ethz.ch/capo/wiki/2012, 2007–2012.

approximate biological performance. For this task we require synthesis methods analogous to the very successful software approach used to program computational processes in general purpose digital computers. However, rather than programming logical operations onto a digital substrate, as done by software compilers, the neuromorphic approach must configure behavioral dynamics onto electronic neural networks. Ideally such a method should provide a means of forward programming the general form of behavioral tasks to be achieved, while allowing the neuronal circuits to optimize their performance through learning and adaptation.

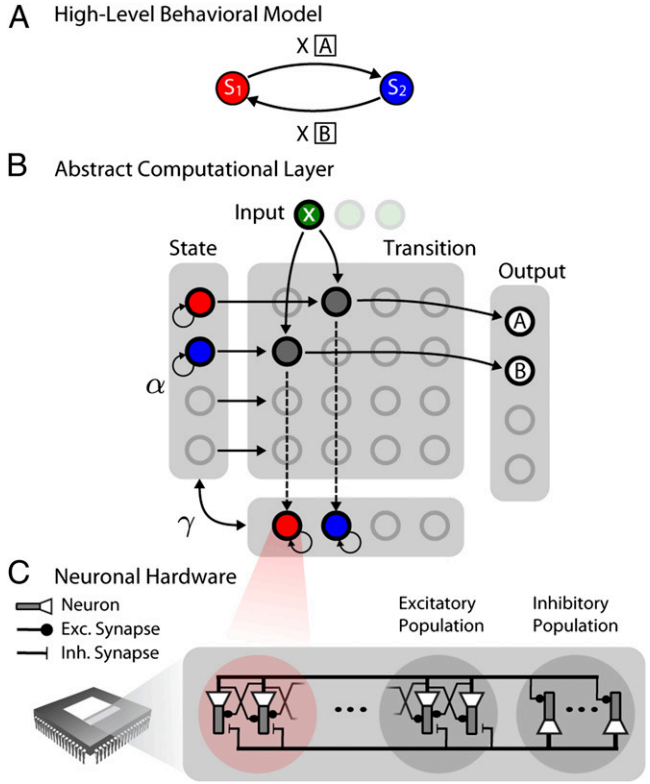

Fig. 1. Synthesis of a target FSM in neuromorphic VLSI neural networks. (*A*) State diagram of the high-level behavioral model. Circles represent states and arrows indicate the transitions between them, conditional on input symbol X. In this example state machine, the active state flips between S_1 and S_2 in response to X and outputs either the response A or the response B, depending on the previous state. (*B*) The computational substrate composed of three sWTA networks: two "state-holding" networks (vertical and horizontal rectangles) and a third transition network (central square). The shaded circles in each sWTA represent populations of spiking neurons that are in competition through a population of inhibitory neurons (not displayed). The state-holding sWTA networks are coupled population-wise (γ-labeled arrow, red with red, blue with blue, etc.) to implement working memory. Solid arrows indicate stereotypic couplings, and the dashed arrows indicate couplings that are specific to the FSM (in this case the one shown in *A*). The gain and threshold in the transition sWTA are configured such that each population becomes active only if both of its inputs are presented together. The sWTA competition ensures that only a single population in the network is active at any time. An additional output sWTA network is connected to the transition network to represent the output symbols. To program a different state machine, only the dashed arrows need to be modified. (*C*) The multineuron chips used in the neuromorphic setup feature a network of low-power I&F neurons with dynamic synapses. The chips are configured to provide the hardware neural substrate that supports the computational architecture consisting of sWTA shown in *B*. Each population of an sWTA network is represented in hardware by a small population of recurrently coupled spiking neurons ($N = 16$), which compete against other populations via an inhibitory population.

Our method for synthesizing cognitive tasks in neuromorphic systems recognizes three conceptual levels of processing: the high-level behavioral model, an intermediate abstract computational layer composed of populations of neurons configured as soft winner-take-all (sWTA) networks (14), and an underlying level of analog–digital neuronal hardware (Fig. 1). We cast the behavioral task as a state machine (9). This behavioral model is then configured on the computational layer by the introduction of relatively sparse connections that connect some neurons of the sWTA modules. Some of these connections are chosen such that the network gives rise to attractors that encode processing states in the form of persistent activity (15), and some additional connections provide transitions between the attractor states conditionally on external input (16). The recurrent connections in the sWTA amplify the signal at each neuron and restore it to an analog memory stored in the network connectivity (17) (Fig. S1). Their gain and signal restoration properties enable sWTA networks to be stably combined to solve a wide range of state-dependent computation tasks (18).

There remains the technical problem of mapping ideal sWTA neuronal network models onto the underlying neuromorphic hardware. The adjustable hardware parameters used for controlling the behavior of the VLSI circuits are currents and voltages that do not have a one-to-one correspondence with the parameter types and values used in the computational model. We solve this problem by an automatic method that involves using a mean-field approach to measuring the steady-state firing rate response of the VLSI neurons, deriving the average neural intracellular current that generated this activity, and then fitting the parameters of the CMOS transistor model to provide the estimated neuronal current–discharge relation. In this way the method generates the necessary mappings between the parameters of the sWTA network models and their electronic counterparts (19).

We demonstrate our overall approach by configuring a multichip system as an agent able to solve a context-dependent task that is comparable in complexity to the tasks used to probe cognition in behavioral studies in awake behaving primates (9).

Results

The multichip agent is composed of five multineuron chips (20, 21) and a silicon retina (22). The sensor and chips communicate by action-potential–like address events (23). The specific cognitive task that the agent is required to solve is shown in Fig. 2: A neuromorphic agent observes a complex visual scene presented on a computer screen. The scene consists of independently moving horizontal and vertical bars. The agent is required to respond differently to these bars according to which of the contexts is in force. The current context is determined by a cue that is shown only transiently and so must be remembered. In the first context the agent must produce response A when it observes a horizontal bar entering the right half of the screen; and in the second context it must produce response B when it observes a vertical bar entering the left half of the screen.

The overall task can be described as a finite state machine (FSM) (9), as shown in Fig. 3. A FSM is an abstract machine that can be in only one of its N_q possible states, but is able to transition between states when it receives an appropriate external input symbol drawn from a predefined alphabet of N_s elements. The FSM state diagram describes the machine as a directed graph whose nodes represent machine states and whose edges indicate the possible transitions between them. The edges are labeled by input symbols that govern the appropriate transition. True FSMs can be efficiently implemented in digital systems that can rely on restored logic and digital encodings. However, the implementation of an analogous machine using neurons requires a different approach in which populations of neurons are first configured as sWTA networks, and then some neurons of these networks are interconnected to embed the necessary states and transition rules (16). Unlike conventional FSMs implemented with bistable digital

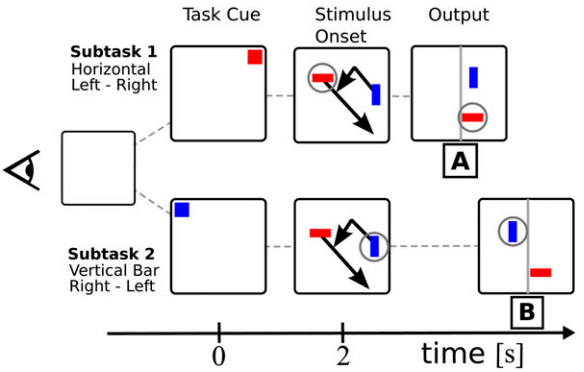

Fig. 2. Context-dependent visual task. Two visual objects, a horizontal and a vertical bar, are moving on a screen and bouncing off its borders. A visual cue flashing at 30 Hz on the upper right corner of the screen for 2 s (red) indicates that the subject must attend to the horizontal bar (indicated by a circle) and report with output A if it enters the right half of the screen. If the initial cue appears on the upper left corner (blue), then the task is inverted: The subject must attend to the vertical bar and report B if the attended bar enters the left half of the screen. The experimental stimuli were presented as black bars against a light background (colors here are used only for the sake of clarity). The agent must respond as soon as the screen midline is judged to be crossed: this fuzzy condition results in different response latencies.

circuits, these neuronal state machines combine analog and digital processing qualities, and so we distinguish them as soft state machines (SSMs). Our goal is not to compete in terms of efficiency with traditional digital systems at the level of the FSM functionality. Rather we present a configuration method that can automatically map ideally specified FSMs into their analogous SSMs composed of silicon neurons, with the perspective of incorporating state-dependent behavior in a large neuromorphic system distributed across multiple chips.

Implementing a Simple Two-State FSM as a SSM. Fig. 1 illustrates, for example, how a simple two-state FSM is mapped into an SSM composed of three coupled sWTA networks. Two sparsely connected sWTA networks (horizontal and vertical rectangles in Fig. 1*B*) provide N_q distinct populations, each able to reliably maintain persistent activity. However, these populations compete with one another via their underlying sWTA dynamics so that only one of the states is expressed (*Materials and Methods*). A third "transition" sWTA network (Fig. 1*B*, *Center*, shaded square) integrates external inputs and state activities to decide the triggering of state transitions. Each population in the transition sWTA network responds to the presentation of exactly one symbol and one state, and the network's winner-take-all functionality permits only one population to be active at any time. The connections between the transition populations and the target state populations are set according to the required FSM state diagram. The SSM input symbols are encoded in an external source of address events, generated on a desktop PC or by another neuromorphic chip. The SSM can produce an output symbol after an input symbol is presented. This is achieved by connecting appropriate transition populations to particular populations in a fourth sWTA that encodes output symbols. The winning population of this output sWTA encodes the current output symbol, which we accept as reporting the agent's responses in the task. In principle, these outputs could be used to drive an explicit motor output such as a robotic arm, or laser pointer, that could physically indicate the agent's response.

The final step of the configuration method is to map the abstract SSM architecture onto the hardware neurons of the VLSI chips by

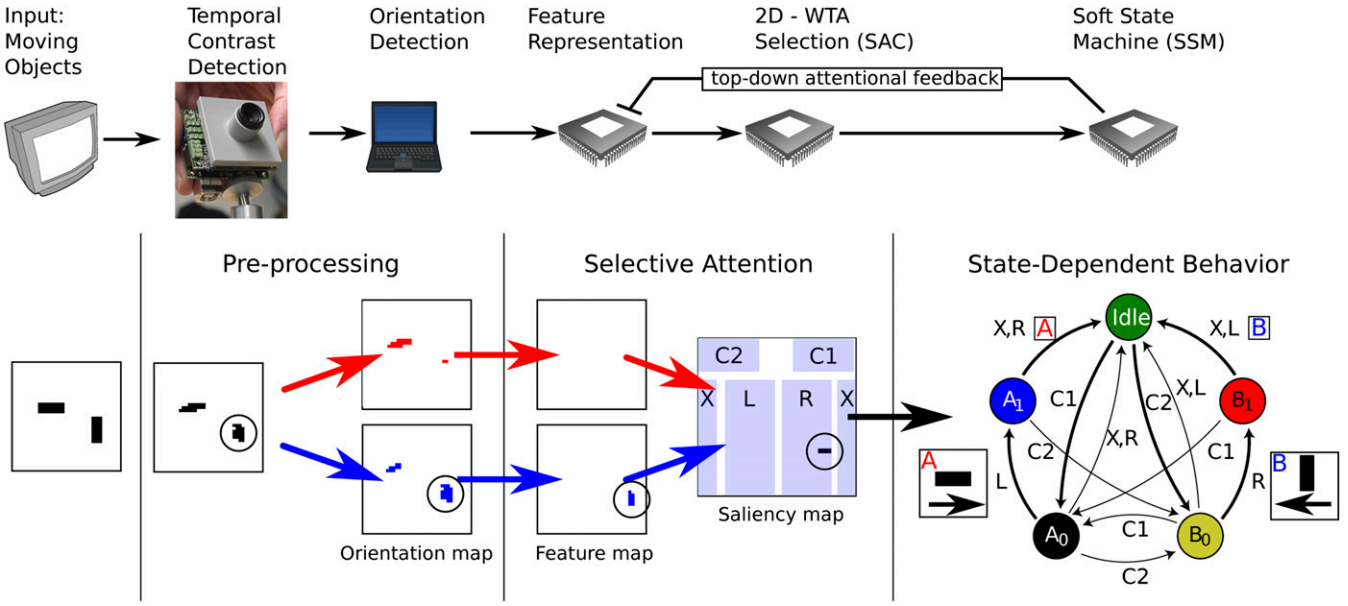

Fig. 3. Real-time neuromorphic agent able to perform the context-dependent visual task. Two moving oriented bars are shown to an event-based 128 × 128 "silicon retina" (22). The silicon retina output events are preprocessed in software to detect orientation and routed accordingly to one of two possible feature maps, implemented as 32 × 32 sheets of VLSI I&F neurons. The events produced by the feature maps are retinotopically mapped to a selective attention chip (SAC), which selects the most salient region of the visual field by activating a spiking neuron at that position (black circle in the Saliency map box). The input–output space of the SAC is divided into five distinct functional regions: left (L), right (R), border (X), and cues (C1, C2). The events from each of these regions are routed to the appropriate transition neurons of the SSM. To focus on the desired target, the system must attend to one of the two bars. This is achieved by modulating the attentional layer with a state-dependent top–down attentional feedback from the SSM. In the neural architecture, this is implemented by inhibiting the features corresponding to the bar that should not be attended to (*Materials and Methods*). Transitions that do not change the state are omitted in the "State-Dependent Behavior" diagram, to avoid clutter. The snapshots shown in the "Pre-processing" and "Selective Attention" diagrams represent experimental data, measured during the experiment of Fig. 4, in the period when the state B0 was active. An additional sWTA network (not displayed) is stimulated by the transition populations to suppress noise and to produce output A or B.

adjusting the neuron circuit bias voltages and by setting the event routing tables that implement the connectivity between the three sWTA networks (*Materials and Methods*). The bias voltages determine the operation of the physical neurons on specific neuromorphic chips and are set only once (independently of the particular SSM that is implemented). The routing tables must be written for each different machine.

Implementing an Agent Able to Perform a Context-Dependent Visual Task. We implemented our chosen cognitive task (Fig. 3, *Lower*, "State-Dependent Behavior") in a neuromorphic system, using the principles outlined in the simple example above. In this task the agent observes a visual scene presented on a computer display screen via a "silicon retina" vision sensor (Fig. 2). The scene contains two independently moving black bars on a light background. The bars move diagonally across the screen and rebound off its borders. The vision sensor generates action potential-like address events when and where its pixels detect local changes in contrast (22) due to the motion of the bars. These asynchronous address events are processed by real-time software (24) that detects local stimulus orientation and assigns a vertical or horizontal orientation label to each event. The events are then transmitted to a multineuron chip comprising an array of 64×32 integrate-and-fire (I&F) neurons. Depending on their orientation and retinal location, they are routed to one of two retinotopically organized orientation feature maps. Each feature map, formed by a 32×32 sheet of VLSI I&F neurons, responds linearly to the excitatory feed-forward input generated by the orientation maps and to the top–down attentional inhibitory feedback input resulting from the SSM computation. The output spikes of the two feature maps converge to the 32×32 input synapses of the selective attention chip (SAC). The net current produced by the SAC synapses represents a saliency map of the visual field. The SAC then selects the region of highest salience, via local winner-take-all circuits, and activates the output neuron corresponding to that location. The SAC input/output space is partitioned into five scene subregions: Left (L), right (R), and border (X) regions represent current positions of stimuli, whereas the cue regions (C1 and C2) indicate which of two contexts is in force. The output events from each of these regions are routed to the appropriate transition neurons of the SSM (Fig. 3, *Lower*, "Saliency Map" box). The SSM processes these symbols according to its embedded state diagram (Fig. 3, *Lower*, State-Dependent Behavior). Thus, the left branch of the SSM recognizes the sequence C1 L R (corresponding to a target moving from the left-hand to the right-hand side of the screen) and generates an output A, whereas its right branch recognizes C2 R L and outputs B. The bars can in principle traverse the midline after reflecting off a screen border. This trajectory results in an invalid stimulus pattern and should be ignored. The SSM achieves this by producing no output and returning to the idle state when the object moves to the border region X.

The agent is required to attend to only one of the two oriented bars, accordingly with the presented cue. Thus, depending on the context, a bar can be considered either a target or a distractor. The selection of the appropriate target is achieved by biasing the competition (25) on the saliency map: The active state in the SSM modifies the saliency map by strongly inhibiting the feature map corresponding to the distractor (Fig. 3, "top–down attentional feedback" pathway). Consequently, the SAC selects the input provided by the noninhibited target feature map.

An example trial is shown in Fig. 4. The agent is in its idle state when the initial context cue C2 is displayed by the experimenter, indicating that the agent should perform the B subtask (attend to the horizontal bar). The C2 stimulus triggers the SSM to switch from idle to B0 (Fig. 4, *Right Middle*). This state encodes both the context in force and which of the two feature maps is inhibited. In this case the activity of B0 inhibits the horizontal feature map via the top–down connections.

On input R, the SSM switches from B0 to B1 (Fig. 4, *Right Middle*, yellow to red spikes in raster plot) while continuing to inhibit the horizontal feature map. When the target in the visual input enters the left half of the screen, the population L of the SAC activates (Fig. 4, *Right Top*, arrow in raster plot). As a result, the SSM produces the correct output (B) and transitions back to idle, eventually releasing the top–down inhibition. Shortly afterward, the experimenter starts a second trial by displaying the C1 cue. The agent successfully completes this trial by generating an output A.

The effect of the top–down attentional feedback can be seen in the snapshots of the feature map activity shown in Fig. 3. In this snapshot the B0 population is active (Fig. 4, *Right Middle*) and so inhibits the horizontal feature map (evident also from the SAC activity in Fig. 4, *Left*), until the agent returns to its idle state. To solve this task, the visual preprocessing subsystem required 3,072 spiking neurons, whereas the SSM required 608 neurons.

Robustness and Scaling. The synthesis method we propose can in principle transform FSMs of arbitrary size into their corresponding SSM (16).

Scaling properties of the state machine are conserved under this transformation. The size of implementation is dominated by the size of the transition mechanisms for both SSMs and FSMs. In the worst case of a full transition matrix, both types of state machines scale as N_q^2. We used a full transition implementation in our system because we wished to evaluate a variety of SSMs by rewiring the same set of modules in our robustness tests. This option means that the state machine has maximum size for the number of states implemented. However, a SSM designed for specific tasks, with fewer possible transitions, would be quite small.

Hardware implementations of state machines can produce errors due to delays and glitches of the logic gates (26). In our SSM implementation, the errors are caused by device mismatch, noise, and finite-size effects. Most SSM errors occurred during state transitions, either because an incorrect state population became active or because the activity of an active state population decayed (i.e., did not remain in its persistent state) due to transient fluctuations in neural activity. The chances of incorrect transitions grow with the size of the SSM because each transition is the result of one population winning the competition between at most $N_s N_q$ others in the sWTA.

We explored whether large and complex physical SSMs could in principle be built using our hardware by analyzing how errors affect the system's ability to process long strings. We did this by synthesizing 10 randomly specified state machines of size $N_q = 5$ and $N_s = 5$, transforming each of these to its corresponding SSM, and then measuring the number of consecutive symbols the SSMs could successfully process without making an incorrect transition (*Materials and Methods*). The curves and their shadings in Fig. 5*A* indicate the mean and the SD of correctly processed string lengths over the collection of random state machines.

The agent is particularly sensitive to the ambiguous situation in which an identical input could induce a transition from both the current state and the new state. These errors are due to an ambiguity as to when the stimulus ends: for example, in the state machine of Fig. 1*A*, an X vs. an XX. In the SSM, the effect of such ambiguous transitions is reduced by matching the input duration to the period of the oscillatory behavior that emerges from the network interactions in the sWTAs (*SI Text*). Despite this countermeasure, ambiguous transitions were less accurate, with a success rate of 88.97%, vs. 95.44% when the transitions were nonambiguous (Fig. 5*B*). The proportion of correct transitions was highest for the self-transition type (97.40%), because the populations implementing them were not wired back to working-memory populations and therefore had little or no effect on the SSM. On average, 93.43% of the transitions we recorded were carried out successfully, which is significantly better than random (Fig. 5*B*, thick line, 20%). The proportion of accurately processed

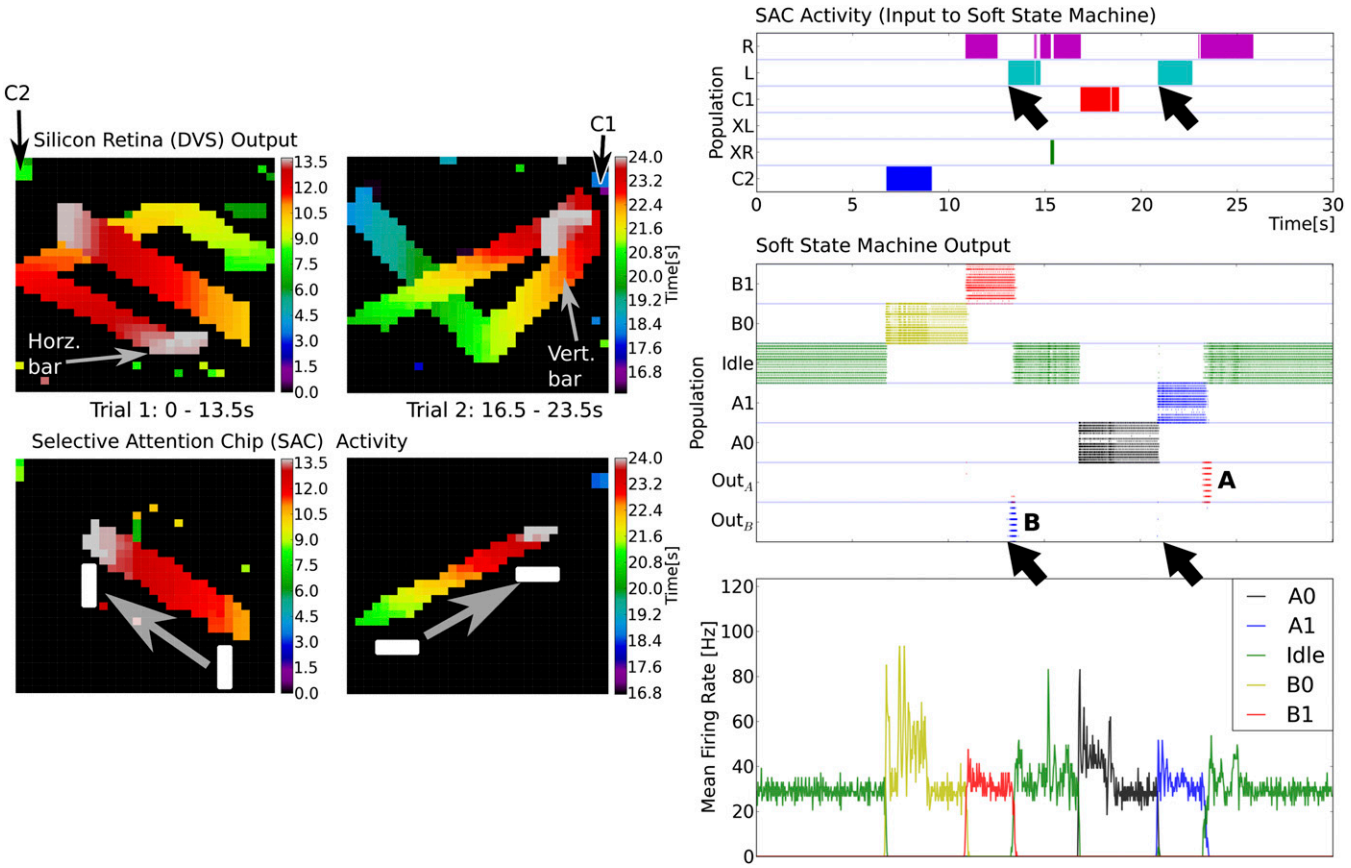

Fig. 4. Results of the visual task experiment. (*Left*) The silicon retina output (*Upper*) and the SAC output (*Lower*). The axes respectively represent the *X-Y* coordinates of the events and the color encodes time. The scattered events around the main stimulus are due to spontaneous activity in the silicon retina. The top–down modulation strongly inhibited the feature map corresponding to the distractor (*Results*). For this reason, in the *Lower* panels, only the target associated to the context in force is observed. The output of the SAC is routed to the corresponding transitions neurons in the SSM (Fig. 3). (*Right Top*) The raster plot of the routed events is shown. The detection of the patterns A and B is reported by the output populations Out_A and Out_B, as shown in the *Right Middle* raster plot. The arrows show a clear example of state-dependent computation: Input L induces either an output B or no output, depending on the context in force. *Right Bottom* plot shows the mean firing rates of the respective populations.

strings decreases exponentially with length (Fig. 5*A*). Randomly generated SSMs without ambiguous transitions could process about 60% of the presented strings on average, whereas this number fell to 20% for SSMs having ambiguous transitions. The SSM designed to solve the task of Fig. 2 could correctly process 82% of strings of length 20 (labeled "Visual task" in Fig. 5*A*). State-dependent tasks are demanding even for humans, who exhibit performances comparable to those achieved by the described agent. For example, in ref. 27 human subjects were able to achieve an accuracy of 98.6% in a working-memory task of similar complexity (requiring five states) after extensive training lasting 800 trials.

These results agree with the robustness tests made for the abstract coupled sWTA networks (16) (where the authors show greater than 90% reliability for four-state machines with moderate noise level) and with the network proposed in ref. 9, which implements a five-state FSM able to execute 95–97% of transitions correctly after extensive training. These results also confirm that the configuration procedure we propose can robustly implement a randomly specified SSM on hardware composed of noisy and imprecise neural circuits, with complexities comparable to those of neurally plausible software models used for solving cognitive tasks (9, 28–31).

Robustness is improved in these networks by reducing fluctuations in the neural activity of the sWTA. This reduction can be achieved by increasing the number of neurons per population or the number of their synapses in each population (32). Other methods for improving robustness include adaptation and learning mechanisms such as those proposed in ref. 31 as well as probabilistic and stochastic mechanisms (33), for example to implement probabilistic inference with Markov models (34).

Discussion

We have described a method for the configuration of simple cognitive behaviors on electronic neuromorphic neural systems and demonstrated a typical cognitive task in which a neuromorphic agent is required to classify in real-time motion patterns, conditional on a contextual signal. This task may appear modest, but it should be noted that similar tasks are used routinely in experiments designed to probe cognition features such as working memory, action selection, and context-dependent sensorimotor mappings in monkeys and humans (11, 28, 35). Of course, such cognitive tasks could in principle be simulated on digital computers using neurally plausible models (31, 36–38). Indeed, Eliasmith et al. have recently demonstrated how some simple cognitive tasks can be systematically compiled onto the connectivity of 2 million simulated spiking neurons (39). In contrast, our goal was to demonstrate that simple behaviors can be implemented in a real-time electronic neuromorphic system composed of asynchronously communicating neurons.

Even in the ideal world of digital simulation, the configuration of simple cognitive tasks is not straightforward. The agent must

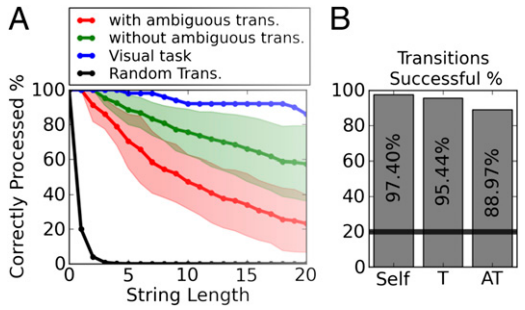

Fig. 5. Robustness of randomly specified SSMs. (*A*) Performance measured as the percentage of correctly processed strings as a function of string length. To emphasize the effect of such errors, we separated the SSMs into two classes, with (red) and without (green) ambiguous transitions. The shaded regions show the SD over a collection of five randomly specified state machines. The blue curve shows the accuracy of the SSM used for the context-dependent visual task (Fig. 3). Each SSM was run with 50 different strings of length of 20. (*B*) Proportion of successful transitions per type of transition, namely self-transitions and ambiguous (AT) and nonambiguous transitions (T), computed from 3,542, 2,717, and 3,417 transition measurements, respectively. The theoretical chance level is computed by assuming arbitrary transitions regardless of the input (thick black line), meaning $\left(\frac{1}{N_Q}\right)^n$.

construct suitable context-dependent sensorimotor mappings. These require mechanisms for working memory, decision making, and action selection. Such mechanisms pose a variety of challenges for artificial neural systems (29–31, 36, 40), and so behavioral tasks are not easily programmed even into abstract models of neuronal computation. If the simulation commits to the neuronal level of operation, the configuration problem becomes even more difficult. For example, Hopfield networks, multilayer perceptron networks, or liquid state machines are powerful computational models that could in principle be used to configure neural hardware to solve state-dependent tasks. However, their task-related architectures emerge through protracted training, and the connection patterns by which these learned networks finally achieve their performance are usually opaque and bring few insights into their possible generalization to other tasks or systems.

In the less ideal world of physical implementation of neurons, the above problems are compounded by the realities of circuit design. Although neuromorphic chips are fabricated using standard CMOS VLSI foundries, their electronic circuits are unlike conventional industrial digital or analog circuits. They are a hybrid of analog and digital technologies, and their analog transistors usually run in the subthreshold regime to emulate very compactly the crucial functional characteristics of biological neurons (41, 42). Consequently, neuromorphic circuits are subject to substantial fabrication variability (device mismatch) and operating noise. Unlike digital circuits, analog circuits require dedicated signal restoration mechanisms to avoid signal corruption through successive stages of processing (3). A similar signal restoration problem holds for biological neurons, as was quickly recognized by von Neumann when he considered alternative architectures for early computers (2).

Why then develop a neuromorphic computational technology beset by such difficulties, in the face of the hugely successful digital approach? The reason is that the reliability of digital systems comes at high cost of circuit and power dissipation. These costs arise out of the need to encode and then simulate numerically the physics of neurons, rather than emulating directly the physics itself, as neuromorphic systems strive to do. This distinction is illustrated very dramatically by comparing the power dissipation of a state of the art supercomputer simulation of say 1 million neurons with a state of the art neuromorphic emulation of 1 million neurons. Modha's recent impressive simulation of 10^{10} neurons running only a hundred times slower than real time on the IBM Bluegene/Q (43) dissipates some 10 MW, whereas a real-time neuromorphic emulation of 10^6 neurons on Boahen's "Neurogrid" dissipates about 3.1 W (44). This comparison indicates that even at the system level neuromorphic implementations are 1,000-fold more efficient (power/neuron) than their digitally simulated counterparts and suggests that significant advantages could accrue by deploying neuromorphic systems for useful processing.

Our approach may appear to be relevant only to systems whose physical implementation layer is composed of poorly specified or unreliable physical components. However, similar concepts to these may become relevant even to digital systems as they scale up to sizes where the conventional model of complete specification, validation, and control is no longer feasible.

Our key insight in dealing with such systems is that the uncertain hardware elements should by simple local mappings contribute to a more computationally stable intermediate layer. In our example, it is this intermediate abstract neural architecture that couples the behavioral model to the hardware (or even biological) neuronal implementation. This intermediate layer serves three purposes. First, it provides computational primitives on which behaviors are easily cast and learned. Second, these primitives provide basic signal processing properties necessary for steering and stabilizing processing such as signal gain, signal restoration (17), and multistability (45). Finally, the layer of primitives hides the details and variability of the neuronal hardware implementation from the behavioral level. Biology may have discovered a similar strategy, for example in the neocortex, which appears to be a sheet of essentially similar local circuitry that can be configured to satisfy a variety of processing tasks (12).

We chose an intermediate abstract neural architecture that is composed of sWTA neural network modules. The sWTA operation reports the strongest subset of a collection of inputs and optionally reproduces them with a specifiable gain (46, 47). The connectivity necessary for constructing sWTAs is consistent with that observed among neurons in the superficial layers of the neocortex (48), and so this circuit has been proposed as a potential neural computational primitive (12) that can be combined easily and stably in large networks that exhibit attractor dynamics (18). Among other functions, these networks can also be configured as SSMs (16, 18). However, these SSMs are unlike their classical digital counterparts, in that they combine digital selection with analog signal processing properties (45), an intriguing duality that we have exploited here. Also the sWTA networks used to implement SSMs differ from digital systems, in that they restore their input signals toward patterns stored in their connections, rather than toward logic levels (Fig. S1). Indeed, the SSM is not simply an imperfect replacement for a conventional FSM based on traditional electronic hardware, but rather a canonical computational primitive for synthesizing complex behaviors in neuromorphic systems.

The ability to synthesize behaviors on the SSM has a number of appealing features: First, cognitive tasks of the kind described here can be designed at the FSM state diagram level, using a variety of computer-aided design tools, which is much more convenient than designing a neural network from scratch. Second, the abstract computational layer, composed of modular sWTA networks, can be easily mapped onto multiple neuromorphic chips. Third, the parameters of the sWTA networks can be reliably and automatically mapped onto the electronic voltage and current biases of the CMOS electronic neurons (19). Finally, only a small number of connections between the transition neurons and the state populations need to be specified or learned to achieve a desired functionality. This property can be useful for framing the design of learning algorithms and for reducing the number of on-chip plastic synapses required to implement autonomous learning of behaviors (7, 49, 50). The complexity of a learning problem is proportional to the dimensionality of the to-be-learned parameters (51). Providing a computational substrate that enables learning of

complex behaviors by modifying only a small number of synapses thus reduces both the time and number of data samples required (13, 52), a procedure analogous to regularization in learning theory.

The state-encoding scheme that is adopted in the SSM can be extended to more biologically plausible, distributed representations. Models of distributed state encodings were shown to be compatible with experimental observations of perceptual dominance (53), position encoding in grid cells (54), context and task rule encodings (31), and taste processing (55). Extensions to the SSM of this sort can enable the neuromorphic implementation of more versatile hidden Markov models (55), which can be used to solve a wide variety of machine-learning tasks (34).

The neuromorphic hardware used in this work to solve the behavioral task comprises multiple prototype chips built to explore general principles of neural computation in hardware. The chips were not designed with this specific application in mind, and the overall system represents a proof of principle. Its success now opens the way for designing a more specific neuromorphic system composed of large fields of sWTA networks for efficient state-dependent sensory-motor processing. The real-time nature of these neuromorphic circuits and networks, together with their compactness and low-power features, will allow their integration into autonomous robotic platforms to augment the event-based architectures that are already being developed (56). In this prototype the visual processing (visual stimulus orientation detection) was performed by real-time address-event processing software (57). In future versions this processing could also be performed in neuromorphic hardware by using simple-cell orientation selectivity hardware models (58, 59) or event-driven convolution chips (60).

Overall, the important insight of this paper is that the abstract layer composed of sWTA provides reliable processing on the unreliable underlying neuromorphic hardware, while simplifying the programming of high-level behavior. This approach is analogous to the manner in which software is used to program and compile computational processes in general purpose digital computers, except that the underlying neuromorphic hardware is radically different from digital ones in both system concept and electronic implementation. The approach is sufficiently general to be used on a wide range of electronic neural networks that have reconfigurable synaptic weights and reprogrammable connectivity (6, 7, 61).

Materials and Methods

Mathematical Tools for Analyzing Networks of VLSI Spiking Neurons. We use a mean-field approach (33, 62, 63) to model the spiking activity of the sWTA networks. In a mean-field model, the state variables represent collective dynamics of the neurons composing the population, via analytically derived activation functions. The mean-field approximation is appropriate when the effect of the synaptic inputs is very small but the overall firing rate is high. For our experimental system, this means (*i*) the charge delivered by each spike to the postsynaptic neuron is small compared with the typical charge necessary to make the neuron spike, (*ii*) there are a large number of afferent inputs to each neuron, and (*iii*) the spike times are uncorrelated. The network parameters were chosen to satisfy these assumptions.

The working memory used to maintain state is supported by self-sustained, persistent activity. Persistently active states can be stable if the dynamics are governed by a slow temporal component, such that the temporal fluctuations are small and the spiking activity in the network is asynchronous (62, 64). In our experiments, the time constants of the excitatory VLSI synapses lie in the "NMDA regime" (50 – 150 ms) (62), and so the dynamics of the neural activity are governed by the synaptic dynamics. Slow time constants in the network result in a "low noise regime", in which the mathematical formulation of the dynamics becomes tractable. In this regime, the firing rate of a spiking neuron population can be approximated with a threshold-linear activation function, $\sigma(\cdot) = \max(\cdot, 0)$ (19, 63). For convenience, we call a population described by a mean-field state variable a linear threshold unit (LTU). The equations describing the dynamics of n LTUs are governed by the synaptic time constant τ,

$$\tau_i \dot{x}_i = x_i + \sigma\left(\sum_{k=0}^{N} w_{ik} x_i - T\right), i = 1, \ldots, n,$$

where w_{ik} is the weight of the connection between LTUs k and i, and T is the threshold due to the neuron's leak. The state variable x describes the "synaptic state", instead of the firing rates, but these two are proportional in steady state (19). In this interpretation, τ_i is equal to the synaptic time constant. The weight is $w = Mpq$, where q is a dimensionless real number representing the synaptic efficacy, M is the number of spiking neurons in a LTU, and p is the probability that the neurons in the LTUs connect to each other. In our implementation, we varied both p and q to configure the weight. The systematic translation between the LTU model and VLSI neuron models is explained in ref. 19.

Configuration of the Soft Winner-Take-All Networks. We briefly describe the mathematical methods underlying the configuration of the sWTA networks to implement the desired SSM. A more detailed discussion can be found in ref. 16. We begin with the stability of a single persistent activity state and then describe the transition between two states. Fig. 1*C* shows the recurrent couplings in the spiking neural network that implements the sWTA. The excitatory neurons in the sWTA network are grouped into subpopulations, each of which is abstracted by one LTU. The excitatory LTUs excite themselves with weight w_E and their respective inhibitory unit with weight w_{EI}. When the inhibitory units activate, they inhibit their respective excitatory units with weight w_{IE}. Let x_E be the activity of an excitatory unit and x_I be the activity of the inhibitory unit. The dynamics of these two units are described by the differential system

$$\begin{aligned} \tau_E \dot{x}_E &= -x_E + \sigma(w_E x_E - w_{IE} x_I - T_E + b), \\ \tau_I \dot{x}_I &= -x_I + \sigma(w_{EI} x_E - T_I), \end{aligned}$$

where T_E and T_I are the thresholds of the excitatory and inhibitory LTUs, respectively; b is the externally applied input; and τ_E and τ_I are the time constants of the excitatory and inhibitory LTUs, respectively. A straightforward linear stability analysis shows that a persistent activity state is stable if (65)

$$\Lambda > 0 \quad \text{and} \quad w_E > \frac{T_E}{T_I} w_{EI} + 1,$$

where $\Lambda = (1 - w_E + w_{EI} w_{IE})$ is the gain of the sWTA circuit and $c = (w_{IE} T_I - T_E)$ (Fig. S4). The firing rate in the active state is $\frac{c}{\Lambda}$.

We now turn to the transition between two arbitrary states, represented by two LTUs, labeled for convenience pre and post. A nullcline analysis (*SI Text*) shows that a transition from pre to post occurs if an input $b > b_{\min} = c\left(\frac{w_{IE} w_{EI}}{\Lambda} - 1\right)$ is provided to post. Two recurrently coupled sWTA networks can be used to maintain the states (16, 18). Each state is then implemented by two populations in the two identical sWTA networks, excitatively coupled to each other, with a coupling of weight γ verifying $w_E = \alpha + \gamma$, where α is the excitatory feedback within each state population (Fig. 1). Using two sWTAs allows us to build the transition sWTA and the state-holding sWTA on the same chip, which would otherwise be difficult due to the limited number of synaptic weights we can distinctively set in our hardware. The analysis remains identical to the single sWTA case, because the states of the two sWTA networks are guaranteed to remain synchronized (18). Therefore, we study the activity of the two state-holding populations using one LTU.

The activation of post is conditional on the activation of pre and the external input. This conditional branching is implemented by a "transition sWTA network". One of its subpopulations, T, activates when these two conditions become true and strongly stimulates post in the state-holding sWTA (Fig. S3). The transition sWTA is configured such that no persistent activity state exists. In addition, we assume that the inhibitory synapses are much faster than the excitatory synapses ($\tau_I \ll \tau_E$), such that x_I can be approximated as steady state ($\tau_I \dot{x}_I \cong 0$), and that the inhibition common to pre and post is always active such that we can omit the threshold function for x_I. The responses of the three LTUs in steady state are then

$$\begin{aligned} x_{\text{pre}} &= \sigma\left(w_E x_{\text{pre}} - w_{IE} w_{EI}\left(x_{\text{pre}} + x_{\text{post}}\right) + c\right), \\ x_{\text{T}} &= \sigma\left(w_{E\text{T}} x_{\text{T}} - c_{\text{T}} + \phi_1 x_{\text{pre}} + b_{\text{in}}\right), \\ x_{\text{post}} &= \sigma\left(w_E x_{\text{post}} - w_{IE} w_{EI}\left(x_{\text{pre}} + x_{\text{post}}\right) + c + \phi_2 x_{\text{T}}\right), \end{aligned}$$

with $c_{\text{T}} = \sigma(w_{IE} w_{EI} x_{\text{T}} + w_{IE} T_I) - T_E$. ϕ_1, ϕ_2 are the synaptic weights between the state-holding sWTA and the transition sWTA, and $w_{E\text{T}}$ is the weight of the excitatory coupling in the transition sWTA.

To drive the transition from pre to post, we exploit the supralinear response of the sWTA: When configured appropriately, the response to the sum of two inputs can be much larger than the sum of the responses to each input taken separately. The external input b_{in} projects to T and verifies $x_T(b_{in}) < b_{min}/\phi_2$, such that post cannot be activated by the input alone. When pre is active, the transition population T receives an input equal to $c/\Lambda\phi_1$. To avoid activating post with the state input alone, the ϕ_1 coupling must verify $x_T(c/\Lambda\phi_1) < b_{min}/\phi_2$. Finally, to trigger a transition to post when both pre and the input are active, x_T must reach at least b_{min}/ϕ_2. This means that b_{in} must drive T to satisfy $\phi_2 x_T > b_{min}$. The parameters used in the neuromorphic setup are provided in Table S1.

The external input consisted of short bursts of spikes to the designated transition neurons. The duration of the burst was manually adjusted around 300 ms once the chip parameters were set, which roughly corresponds to the period of the oscillatory behavior that emerges from the network interactions in the sWTAs (*SI Text* and Fig. S2). The output of the SSM is implemented by projecting the transition population activities or the state-holding population activities to subpopulations in a fourth sWTA network.

Neuromorphic Hardware and Configuration Method. The multineuron chip used in the neuromorphic setup features a network of low-power I&F neurons with dynamic synapses. The input and output communication of spiking activity of the neurons is encoded using the address-event representation (AER) protocol (66). In AER, digital events produced by the spiking neurons are transmitted through a single digital communication pathway via time multiplexing. The chip can be interfaced to a workstation through dedicated boards that allow one to stimulate the synapses on the chip and monitor the spiking activity of the neurons.

The SSM consists of three recurrently connected sWTA networks, coupled to each other through an AER mapper board (67). The neural circuit implementing the sWTA is illustrated in Fig. 1*C*. The three sWTAs are implemented in a multichip setup composed of two chips: one providing up to 2,048 excitatory neurons and another providing up to 128 inhibitory neurons. The 2,048-neuron chip contains 32 networks of 64 neurons, each tied to three differential-pair integrator synapses (68) that are stimulated by externally provided address events. In addition, there are local hardwired couplings between the first and second nearest neighbors and self-excitation synapses (69). The lateral excitatory connections were designed to implement the cooperation in a one-dimensional sWTA operation. The 128-neuron chip is of similar design and was used to implement the global inhibition (20). Both chips have been fabricated using a standard 0.35 μm CMOS process and cover areas of about 17 mm^2 (2,048 neurons) and 10 mm^2 (128 neurons), respectively.

The SSM can produce an output after each incoming symbol. The output is achieved by connecting appropriate transition populations to particular populations in a fourth sWTA that encodes the output symbols. The output sWTA network was implemented by a 128-neuron chip that includes hardwired excitatory and inhibitory couplings (20).

Our configuration method maps ideal FSMs into their analogous SSMs composed of silicon neurons and distributed across multiple neuromorphic chips. All implemented state-dependent behaviors were first specified as Mealy machines, using the Java Formal Languages and Automata Package (JFLAP) software (70). Custom-written Python-based software (71) translated the XML files produced by JFLAP into the event-routing tables and chips biases used to configure the SSMs. The size of the sWTA was set according to the number of attractor states (N_q), inputs (N_s), and neurons per LTU.

For SSMs whose output symbols do not explicitly depend on its input symbols (i.e., similar to Moore machines or automatons), the transition sWTA emulates a transition matrix with possibly all nonzero entries full transition matrix. Therefore, the required number of neurons scales as $N_{exc}(2N_q + N_q^2) + 3N_{inh}$, where N_{exc} and N_{inh} are the number of excitatory and inhibitory neurons per LTU, respectively. If the output symbol depends on both state and input symbol (i.e., similar to a Mealy machine), then the SSM requires up to $N_{exc}(2N_q + N_q \cdot N_s) + 3N_{inh}$ neurons. For all our experiments, we chose $N_{exc} = N_{inh} = 16$. Similarly, the number of synaptic events scaled as $N_s N_q$. For a randomly specified SSM of dimensions $N_q = 5, N_s = 5$, we observed about 80,000 synaptic events per second when no input was provided, and this number increased to 300,000 during transitions (Fig. S5). The parameters of the sWTA networks were initially configured using the methods described in ref. 19. A calibration mechanism described in ref. 72 was used to decrease the effect of fabrication mismatch in the transition sWTA. All of the reported neural firing rates were computed using 30 − ms time bins, averaged over all of the neurons in a population.

Robustness Measurements in Randomly Specified SSMs. The robustness of the SSM is assessed by its ability to correctly process long sequences of symbols. A random state machine is constructed by randomly selecting the target state of the transitions. The state machine generation method was based on the methods used in ref. 16. The state of the system was determined by observing which population's activity in the state-holding sWTA exceeded 15 Hz, 500 ms after the end of the input stimulus. The state machine was initialized to its predefined initial state by direct 300 − ms stimulation. The input consisted of 20 consecutive symbols that were separated by 700 − ms intervals. The state switch was considered successful if the system switched to the correct state and had an activity above 15 Hz. This experiment was repeated 50 times for each SSM. The outcome is analyzed in two ways. In the first analysis, we computed X_s^n, where $X_s^n = 1$ if the first n transitions of string s were correctly processed and 0 otherwise. We reported the number of consecutive symbols the SSM had correctly processed, $Y^n = \sum_{s=1}^{50} X_s^n/50$ for $n = 1, \cdots, 20$. For a perfect SSM, $Y^n = 1, \forall n$, and for a state machine performing at chance level, meaning a transition to an arbitrary state regardless of the input, $Y^n = \left(\frac{1}{N_q}\right)^n$, with N_q the number of states. In the second analysis, we generated a database of all of the transitions of all of the 10 randomly generated state machines (totaling 9,677 transitions) and computed the proportion of successful transitions. These were then reported separately for each type of transition (self-transitions and ambiguous and nonambiguous transitions).

Visual Input and Attentional Preprocessing. The stimuli were generated with VisionEgg Software (73). The events produced by the 128 × 128 Dynamic Vision Sensor (DVS) camera were preprocessed to detect orientation, using an open-source real-time event-processing program, jAER (57) (using the built-in SimpleOrientationFilter). The visual events were labeled by the detected orientation and routed retinotopically to one of the two feature maps through a network socket. The feature maps consisted of 32 × 32 VLSI I&F neurons that responded linearly to their inputs. Each neuron of the feature map was receptive to distinct 4 × 4 patches of DVS neurons. The feature map neurons were implemented by the same chip used for the excitatory neurons in the SSM. The events produced in the feature maps were routed to a SAC (21). The top–down feedback inhibited the feature map related to the distracting target defined by the ongoing subtask, such that the activity provided by the nonsuppressed feature map won the competition in the SAC. In our case, the left branch of the SSM (states A0 and A1) inhibited the horizontal feature map, whereas the right branch (states B0 and B1) inhibited the vertical feature map. The SSM allowed the state to switch from one subtask to the other even while it was not in idle, despite the top–down inhibition. This is because the cue inputs C1 and C2 are not oriented such that their activity is not completely suppressed by the top–down feedback. The implementation of the SSM required 608 neurons ($N_q = 5$, $N_s = 5$, $N_{exc} = N_{inh} = 16$). The SAC and the chip implementing the feature map were part of a second neuromorphic setup that communicated bidirectionally with the SSM setup through a network socket.

ACKNOWLEDGMENTS. We thank Daniel Fasnacht for designing the AER infrastructure; Chiara Bartolozzi for the SAC; Tobi Delbruck for discussion and jAER support; Tobi Delbruck and Patrick Lichtsteiner for the DVS camera; Shih-Chii Liu, Jean-Jacques Slotine, and Matthew Cook for discussion; and Michael Pfeiffer and Florian Jug for review. This work was supported by the European Union (EU) European Research Council Grant "neuroP" (257219), by the EU Information and Communication Technologies Grant "acoustic SCene ANalysis for Detecting Living Entities (SCANDLE)" (231168), and by the Excellence Cluster 227 (Cognitive Interaction Technology–Center of Excellence, Bielefeld University).

1. Mead CA (1989) *Analog VLSI and Neural Systems* (Addison-Wesley, Reading, MA).
2. von Neumann J (1958) *The Computer and the Brain* (Yale Univ Press, New Haven, CT).
3. Sarpeshkar R (1998) Analog versus digital: Extrapolating from electronics to neurobiology. *Neural Comput* 10(7):1601–1638.
4. Indiveri G, Horiuchi TK (2011) Frontiers: Frontiers in neuromorphic engineering. *Front Neurosci* 5, 10.3389/fnins.2011.00118.
5. Seo J, et al. (2011) A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. *Custom Integrated Circuits Conference (CICC)* (Institute of Electrical and Electronic Engineers, New York), pp 1–4.
6. Silver R, Boahen K, Grillner S, Kopell N, Olsen KL (2007) Neurotech for neuroscience: Unifying concepts, organizing principles, and emerging tools. *J Neurosci* 27(44):11807–11819.
7. Schemmel J, et al. (2010) A wafer-scale neuromorphic hardware system for large-scale neural modeling. *International Symposium on Circuits and Systems, ISCAS 2010* (Institute of Electrical and Electronic Engineers), pp 1947–1950.
8. Furber S, et al. (2012) Overview of the spinnaker system architecture. *IEEE Trans Comput* 99:1.
9. Dayan P (2008) Simple substrates for complex cognition. *Front Neurosci* 2(2):255–263.
10. Asaad WF, Rainer G, Miller EK (2000) Task-specific neural activity in the primate prefrontal cortex. *J Neurophysiol* 84(1):451–459.

11. Mansouri FA, Matsumoto K, Tanaka K (2006) Prefrontal cell activities related to monkeys' success and failure in adapting to rule changes in a Wisconsin Card Sorting Test analog. *J Neurosci* 26(10):2745–2756.
12. Douglas RJ, Martin KAC (2004) Neuronal circuits of the neocortex. *Annu Rev Neurosci* 27:419–451.
13. Bousquet O, Elisseeff A (2002) Stability and generalization. *J Mach Learn Res* 2:499–526.
14. Yuille AL, Geiger D (2003) *Winner-Take-All Networks* (MIT Press, Cambridge, MA), pp 1228–1231.
15. Amari S, Arbib MA (1977) Competition and cooperation in neural nets. *Systems Neuroscience*, ed Metzler J (Academic, London), pp 119–165.
16. Rutishauser U, Douglas RJ (2009) State-dependent computation using coupled recurrent networks. *Neural Comput* 21(2):478–509.
17. Douglas RJ, Mahowald MA, Martin KAC (1994) Hybrid analog-digital architectures for neuromorphic systems. *Proceedings of the IEEE World Congress on Computational Intelligence* (Institute of Electrical and Electronic Engineers, New York), Vol 3, pp 1848–1853.
18. Rutishauser U, Douglas RJ, Slotine JJ (2010) Collective stability of networks of winner-take-all circuits. *Neural Comput* 23(3):735–773.
19. Neftci E, Chicca E, Indiveri G, Douglas R (2011) A systematic method for configuring VLSI networks of spiking neurons. *Neural Comput* 23(10):2457–2497.
20. Indiveri G, Chicca E (2011) A VLSI neuromorphic device for implementing spike-based neural networks. *Neural Nets WIRN11 - Proceedings of the 21st Italian Workshop on Neural Nets*, eds Bassis S, Esposito A, Morabito CF, Apolloni B (IOS Press, Amsterdam), Vol 234, pp 305–316.
21. Bartolozzi C, Indiveri G (2009) Selective attention in multi-chip address-event systems. *Sensors (Basel)* 9(7):5076–8098.
22. Lichtsteiner P, Posch C, Delbruck T (2006) A 128×128 120dB 30mW asynchronous vision sensor that responds to relative intensity change. *IEEE ISSCC Digest of Technical Papers* (IEEE International, New York), pp 508–509.
23. Deiss SR, Douglas RJ, Whatley AM (1998) A pulse-coded communications infrastructure for neuromorphic systems. *Pulsed Neural Networks*, eds Maass W, Bishop CM (MIT Press, Cambridge, MA), pp 157–178.
24. The jAER open source project (2006) SourceForge web-site. Available at http://sourceforge.net/projects/jaer. Accessed July 3, 2013.
25. Desimone R, Duncan J (1995) Neural mechanisms of selective visual attention. *Annu Rev Neurosci* 18:193–222.
26. Kohavi Z (1979) *Switching and Finite Automata Theory* (Cambridge Univ Press, Cambridge, UK), 3rd Ed.
27. Krueger KA (2011) Sequential learning in the form of shaping as a source of cognitive flexibility. PhD thesis (University College London, London).
28. Koechlin E, Basso G, Pietrini P, Panzer S, Grafman J (1999) The role of the anterior prefrontal cortex in human cognition. *Nature* 399(6732):148–151.
29. Fusi S, Asaad WF, Miller EK, Wang XJ (2007) A neural circuit model of flexible sensorimotor mapping: Learning and forgetting on multiple timescales. *Neuron* 54(2):319–333.
30. O'Reilly RC, Frank MJ (2006) Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Comput* 18(2):283–328.
31. Rigotti M, Ben Dayan Rubin DD, Wang X-J, Fusi S (2010) Internal representation of task rules by recurrent dynamics: The importance of the diversity of neural responses. *Front Comput Neurosci*, 10.3389/fncom.2010.00024.
32. Amit DJ, Brunel N (1997) Dynamics of a recurrent network of spiking neurons before and following learning. *Network* 8(4):373–404.
33. Martí D, Deco G, Mattia M, Gigante G, Del Giudice P (2008) A fluctuation-driven mechanism for slow decision processes in reverberant networks. *PLoS ONE* 3(7):e2534.
34. Bishop CM (2006) *Pattern Recognition and Machine Learning* (Springer, Secaucus, NJ).
35. Asaad WF, Rainer G, Miller EK (1998) Neural activity in the primate prefrontal cortex during associative learning. *Neuron* 21(6):1399–1407.
36. Frank MJ, Loughry B, O'Reilly RC (2001) Interactions between frontal cortex and basal ganglia in working memory: A computational model. *Cogn Affect Behav Neurosci* 1(2):137–160 10.3758/CABN.1.2.137.
37. Dayan P (2007) Bilinearity, rules, and prefrontal cortex. *Front Comput Neurosci*, 10.3389/neuro.10.001.2007.
38. Heinzle J, Hepp K, Martin KAC (2007) A microcircuit model of the frontal eye fields. *J Neurosci* 27(35):9341–9353.
39. Eliasmith C, et al. (2012) A large-scale model of the functioning brain. *Science* 338 (6111):1202–1205.
40. Daw ND, Niv Y, Dayan P (2005) Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat Neurosci* 8(12):1704–1711.
41. Liu S-C, Kramer J, Indiveri G, Delbrück T, Douglas R (2002) *Analog VLSI: Circuits and Principles* (MIT Press, Cambridge, MA).
42. Indiveri G, et al. (2011) Neuromorphic silicon neuron circuits. *Front Neurosci* 5:73.
43. Preissl R, et al. (2012) Compass: A scalable simulator for an architecture for cognitive computing. *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC 2012)*, ed Hollingsworth JK (IEEE Computer Society Press Los Alamitos, CA), pp 1–11.
44. Choudhary S, et al. (2012) Silicon neurons that compute. *ICANN* 2012:121–128.
45. Hahnloser R, Sarpeshkar R, Mahowald MA, Douglas RJ, Seung S (2000) Digital selection and analog amplification co-exist in an electronic circuit inspired by neocortex. *Nature* 405(6789):947–951.
46. Feldman JA, Ballard DH (1982) Connectionist models and their properties. *Cogn Sci* 6(3):205–254.
47. Grossberg S (1973) Contour enhancement, short term memory, and constancies in reverberating neural networks. *Stud Appl Math* 52(3):213–257.
48. Binzegger T, Douglas RJ, Martin KA (2004) A quantitative map of the circuit of cat primary visual cortex. *J Neurosci* 24(39):8441–8453.
49. Mitra S, Fusi S, Indiveri G (2009) Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans Biomed Circuits Syst* 3(1):32–42.
50. Giulioni M, Pannunzi M, Badoni D, Dante V, Del Giudice P (2009) Classification of correlated patterns with a configurable analog VLSI neural network of spiking neurons and self-regulating plastic synapses. *Neural Comput* 21(11):3106–3129.
51. Vapnik VN (1982) *Estimation of Dependences Based on Empirical Data* (Springer, New York).
52. Kearns M, Ron D (1999) Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Comput* 11(6):1427–1453.
53. Gigante G, Mattia M, Braun J, Del Giudice P (2009) Bistable perception modeled as competing stochastic integrations at two levels. *PLoS Comput Biol* 5(7):e1000430.
54. Sreenivasan S, Fiete I (2011) Grid cells generate an analog error-correcting code for singularly precise neural computation. *Nat Neurosci* 14(10):1330–1337.
55. Jones LM, Fontanini A, Sadacca BF, Miller P, Katz DB (2007) Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proc Natl Acad Sci USA* 104(47):18772–18777.
56. Bartolozzi C, et al. (2011) Embedded neuromorphic vision for humanoid robots. *The Seventh IEEE Workshop on Embedded Computer Vision, ECVW 2011* (Institute of Electrical and Electronic Engineers, New York), pp 1–7.
57. Delbruck T (2008) Frame-free dynamic digital vision. *Proceedings of the International Symposium on Secure-Life Electronics*, ed Hotate K (Univ of Tokyo, Tokyo) Vol 1, pp 21–26.
58. Chicca E, Lichtsteiner P, Delbruck T, Indiveri G, Douglas RJ (2006) Modeling orientation selectivity using a neuromorphic multi-chip system. *International Symposium on Circuits and Systems, ISCAS 2006* (Institute of Electrical and Electronic Engineers, New York), pp 1235–1238.
59. Vogelstein RJ, Mallik U, Vogelstein JT, Cauwenberghs G (2007) Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans Neural Netw* 18(1):253–265.
60. Camunas-Mesa L, et al. (2012) An event-driven multi-kernel convolution processor module for event-driven vision sensors. *Solid-State Circuits IEEE Jf* 47(2):504–517.
61. Arthur JV, et al. (2012) Building block of a programmable neuromorphic substrate: A digital neurosynaptic core. *International Joint Conference on Neural Networks, IJCNN 2012* (Institute of Electrical and Electronic Engineers), pp 1946–1953.
62. Wang XJ (1999) Synaptic basis of cortical persistent activity: The importance of NMDA receptors to working memory. *J Neurosci* 19(21):9587–9603.
63. Fusi S, Mattia M (1999) Collective behavior of networks with linear (VLSI) integrate-and-fire neurons. *Neural Comput* 11(3):633–652.
64. Brunel N (2000) Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J Comput Neurosci* 8(3):183–208.
65. Neftci E, Chicca E, Cook M, Indiveri G, Douglas RJ (2010) State-dependent sensory processing in networks of vlsi spiking neurons. *International Symposium on Circuits and Systems, ISCAS 2010* (Institute of Electrical and Electronic Engineers, New York), pp 2789–2792.
66. Lazzaro J, Wawrzynek J, Mahowald M, Sivilotti M, Gillespie D (1993) Silicon auditory processors as computer peripherals. *IEEE Trans Neural Netw* 4(3):523–528.
67. Fasnacht DB, Indiveri G (2011) A PCI based high-fanout AER mapper with 2 GiB RAM look-up table, 0.8 μs latency and 66 mhz output event-rate. *Conference on Information Sciences and Systems, CISS 2011* (Johns Hopkins University, Baltimore), pp 1–6.
68. Bartolozzi C, Indiveri G (2007) Synaptic dynamics in analog VLSI. *Neural Comput* 19 (10):2581–2603.
69. Chicca E, Indiveri G, Douglas RJ (2007) Context dependent amplification of both rate and event-correlation in a VLSI network of spiking neurons. *Advances in Neural Information Processing Systems*, eds Schölkopf B, Platt JC, Hofmann T (Neural Information Processing Systems Foundation, MIT Press, Cambridge, MA), Vol 19, pp 257–264.
70. Rodger SH, Finley TW (2006) *JFLAP - An Interactive Formal Languages and Automata Package* (Jones & Bartlett Learning, Sudbury, MA).
71. Sheik S, Stefanini F, Neftci E, Chicca E, Indiveri G (2011) Systematic configuration and automatic tuning of neuromorphic systems. *International Symposium on Circuits and Systems, ISCAS 2011* (Institute of Electrical and Electronic Engineers, New York), pp 873–876.
72. Neftci E, Indiveri G (2010) A device mismatch reduction method for VLSI spiking neural networks. *Biomedical Circuits and Systems Conference BIOCAS 2010* (Institute of Electrical and Electronic Engineers, New York), pp 262–265.
73. Straw AD (2008) Vision egg: An open-source library for realtime visual stimulus generation. *Front Neuroinform* 2:4.