A Digital Neurosynaptic Core Using Embedded Crossbar Memory with 45pJ per Spike in 45nm

Paul Merolla[1], John Arthur[1], Filipp Akopyan[1,2], Nabil Imam[2], Rajit Manohar[2], Dharmendra S. Modha[1]
[1]IBM Research - Almaden, [2]Cornell University

***Abstract*—The grand challenge of neuromorphic computation is to develop a flexible brain-like architecture capable of a wide array of real-time applications, while striving towards the ultra-low power consumption and compact size of the human brain—within the constraints of existing silicon and post-silicon technologies. To this end, we fabricated a key building block of a modular neuromorphic architecture, a *neurosynaptic core*, with 256 digital integrate-and-fire neurons and a 1024×256 bit SRAM crossbar memory for synapses using IBM's 45nm SOI process. Our fully digital implementation is able to leverage favorable CMOS scaling trends, while ensuring one-to-one correspondence between hardware and software. In contrast to a conventional von Neumann architecture, our core tightly integrates computation (neurons) alongside memory (synapses), which allows us to implement efficient fan-out (communication) in a naturally parallel and event-driven manner, leading to ultra-low active power consumption of 45pJ/spike. The core is fully configurable in terms of neuron parameters, axon types, and synapse states and is thus amenable to a wide range of applications. As an example, we trained a restricted Boltzmann machine offline to perform a visual digit recognition task, and mapped the learned weights to our chip.**

I. Introduction

The human brain is capable of amazing feats of perception, action, and cognition, and yet consumes less power and space than a personal computer (20W in 2L). While it has become possible to simulate large-scale brain-like networks using modern day supercomputers [1], these simulations are many orders of magnitude less efficient in power and space than biological systems. The promise of neuromorphic computing is to bridge this efficiency gap by vertically integrating across technology (novel nano-devices as well as emerging CMOS processes), design and engineering (scalable neuromorphic architectures), software (hardware-implementable neural algorithms), applications (virtual and real environments), while using neuroscience as the guiding inspiration and mathematics as grounding.

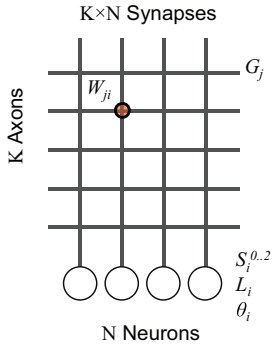
Traditionally, neuromorphic architectures have used analog neurons to significantly reduce power consumption [2]. However, the approach has two main drawbacks. First, there is limited correspondence between software (the neural algorithm) and hardware (the analog implementation) due to the sensitivity of dense analog circuits to process variations and to ambient temperatures. In sharp contrast, our architecture is completely deterministic by design, and hence, functionally equivalent to software models. In particular, for any time step in the software simulation, the software activity in terms of spikes per neuron, neuron state, etc. is identical to the equivalent property of the hardware. This is the first neuromorphic architecture to exhibit this strong correspondence property, breaking a key stalemate between hardware development and software development. Second, our digital approach alleviates some of the scaling challenges of analog neuromorphic systems. In particular, the lack of high-density capacitors in modern CMOS processes as well as increasing sub-threshold currents makes it difficult to implement neural circuits with biologically-relevant time constants.

Given that our digital hardware is equivalent to a software model, one can ask: why not take the software model itself and translate it into hardware directly? This would correspond to an ASIC implementation of the software simulator. Unfortunately this leads to a highly inefficient implementation, because the software has been written assuming a von Neumann model of computation. Specifically, the von Neumann architecture separates memory and computation, and therefore requires high-bandwidth to communicate spikes to off-chip routing tables, leading to high power consumption. Furthermore, the parallel and event-driven computation of the brain does not map well to the sequential processing model of conventional computers. In sharp contrast, we implement fanout by integrating crossbar memory with neurons to keep data movement local, and use an asynchronous event-driven design where each circuit evaluates in parallel and without any clock, dissipating power only when absolutely necessary [3]. These architectural choices lead to dense integrated synapses while delivering ultra-low active power and guaranteeing real-time performance.

As our main contribution, we demonstrate a *neurosynaptic core* that (i) incorporates central elements from neuroscience (digital leaky integrate-and-fire neurons, axons which are output lines of neurons, and synapses that are junctions between neurons); (ii) is scalable with developments in CMOS technology; (iii) implements a new architecture that integrates computation, communication, and memory; and (iv) maintains one-to-one correspondence with software. Our prototype chip consists of a single core with 256 digital neurons, 1024 individually addressable axons that can be excitatory or inhibitory, and 1024×256 programmable binary synapses implemented with a SRAM crossbar array. The entire core fits in a 4.2mm^2 footprint in IBM's 45nm SOI process and consumes 45pJ per spike in active power.

II. Neurosynaptic core specification

The basic building blocks of our neurosynaptic core are axons, synapses, and neurons (Fig. 1). Within the core, information flows from axons to neurons modulated by synapses.



Name	Description	Range
W_{ji}	Connection between axon j and neuron i	0,1
G_j	Axon type	0,1, 2
$S_i^{0..2}$	Synapse values	-256 to 255
L_i	Leak	-256 to 255
θ_i	Threshold	1 to 256

Fig. 1. The core consists of axons, represented as rows; dendrites, represented as columns; synapses, represented as row–column junctions; and neurons that receive inputs from dendrites. The parameters that describe the core have integer ranges as indicated.

The structure of the core consists of K axons that connect via K × N binary-valued synapses to N neurons. We denote the connection between axon j and neuron i as W_{ji}.

The dynamics of the core is driven by a discrete time step that is used to implement a discrete-event simulation. Let t denote the index that denotes the current time step. Typically, t has units of milliseconds.

Each axon corresponds to a neuron's output that could either reside on the core or somewhere else in a large system with many cores. At each time step t, each axon j is presented with an activity bit $A_j(t)$ that represents whether its corresponding neuron fired in the previous time step. Axon j is statically designated as one of three types G_j, which assumes a value of 0, 1, or 2; these types are used to differentiate connections (e.g., excitatory or inhibitory) with different efficacies. Correspondingly, neuron i weighs synaptic input from axon j of type $G_j \in \{0, 1, 2\}$ as $S_i^{G_j}$. Thus, neuron i receives the following input from axon j:

$$A_j(t) \times W_{ji} \times S_i^{G_j}.$$

For our neurons, we use a leaky integrate-and-fire model (single compartment) parameterized by its membrane potential $V(t)$, leak L, threshold θ, and three synaptic values S^0, S^1, S^2 that correspond to the different axon types. The membrane potential of neuron i is updated in each time step as

$$V_i(t+1) = V_i(t) + L_i + \sum_{j=1}^{\mathsf{K}} \left[A_j(t) \times W_{ji} \times S_i^{G_j} \right].$$

When $V(t)$ exceeds its threshold θ, the neuron produces a spike and its potential is reset to 0. We also enforce that negative membrane potentials are clipped back to 0 at the end of each time step.

III. Event-driven implementation

To implement the above specification in hardware, we are presented with non-trivial tradeoffs between power, performance, and density. For our current design, we chose to minimize active power consumption, meet (or exceed) real-time performance, but did not aggressively pursue density optimizations. First, our strategy to reduce active power is to perform neural updates in an event-driven manner. Specifically, we followed an asynchronous design style, where every block performs request-acknowledge handshakes to perform quasi-delay insensitive communication (no clock). Next, our strategy to meet and exceed real-time performance is to implement all neurons using dedicated circuits (non-multiplexed), allowing all the neural updates to happen in parallel. The cost of this extreme parallelism, however, is relative density inefficiency, which is in part mitigated by use of a dense technology.

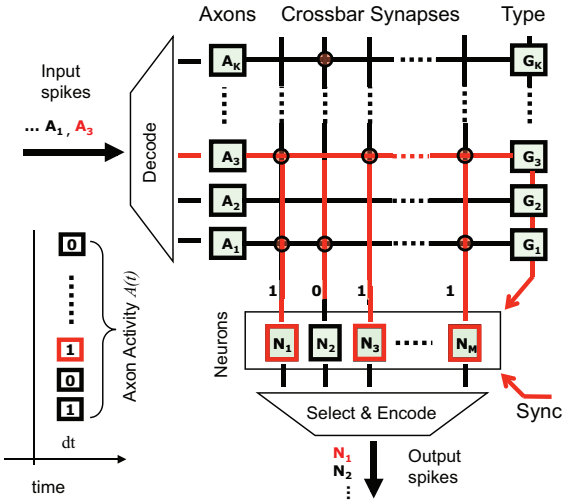


Fig. 2. Internal blocks of the core include axons (A), crossbar synapses implemented with SRAM, axon types (G), and neurons (N). An incoming address event activates axon 3, which reads out that axon's connections, and results in updates for neurons 1, 3 and M.

Based on these considerations, we arrived at a block-level implementation of our neurosynaptic core that consists of an input decoder with 1024 axon circuits, a 1024 × 256 SRAM crossbar, 256 neurons, and an output encoder (Fig. 2). Communication at the input and output of the core follows an *address-event representation* (AER), which encodes binary activity, such as $A(t)$, by sending the locations of active elements via a multiplexed channel [4]. For each time step, the detailed operation of the core commences in two phases: the first phase implements the axon-driven component, and the second phase implements a time step synchronization.

In the first phase, address-events are sent to the core one at a time, and these events are sequentially decoded to the appropriate axon block (e.g., axon 3 from Fig. 2). On receiving an event, the axon activates the SRAM'S row, which reads out all of the axon's connections as well as its type. All the connections that exist (all the `1`'s) are then sent to their respective neurons, which perform the appropriate membrane potential updates; the `0`'s are ignored. After the completion of all the neuron updates, the axon block de-asserts its read, and is ready to process the next incoming address event; this continues until all address events for the current time step are serviced.

In the second phase, which occurs once every millisecond, a synchronization event (Sync) is sent to all the neurons. On receiving this synchronization, each neuron checks to see if its

membrane potential is above threshold, and if so, it produces a spike and resets the membrane potential to 0; these spikes are encoded and sent off as address events in a sequential fashion. After checking for a spike, the leak is applied.

The purpose of breaking neural updates into two phases is to ensure that the hardware and software are always in lock step at the end of each time step. Specifically, the order in which address events arrive to the core or exit the core can vary from chip to chip due to resource arbitration—especially when events are sent through a non-deterministic routing network. To remain one-to-one, the different orderings must not influence the spiking dynamics. We achieve one-to-one equivalence by first accounting for all the axon inputs, and then checking for spikes after these inputs have been processed. This also gives us a precise bound for operating in real time: all address events must be accounted for before the synchronization event, which we trigger once per millisecond.

IV. Test Results

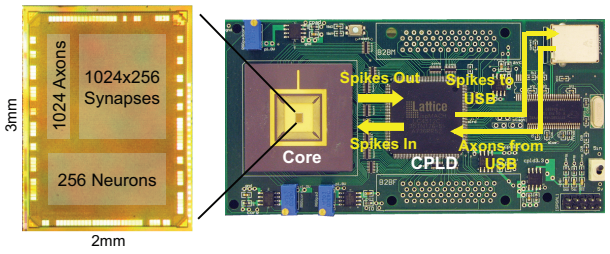


Fig. 3. (left) Neurosynaptic die measures 2mm × 3mm including the I–O pads. (right) Test board that interfaces with the chip via a USB 2.0 link. Spike events are sent to a PC for data collection, and are also routed back to the chip to implement recurrent connectivity.

We have designed, fabricated, and tested our neurosynaptic core using IBM's 45nm SOI process. Our core has 3.8 million transistors in 4.2mm^2 of silicon (Fig. 3, left), and all transistors are ultra-high threshold (UVT) to reduce leakage.

The core's 256 neurons are organized into a 16 × 16 array, where each neuron occupies 35μm × 95μm. For crossbar synapses, we use a custom SRAM array with 1024×256 bits implementing over 256K binary synapses. The bitcell occupies 1.3μm^2 (plus another 1.9μm^2 associated with conservatively-designed periphery). Because our bitcell was custom, its area is approximately 4× larger than the commercially available bitcells in the same technology.

To test our design, we built a custom printed circuit board that interfaces with a PC through a USB link (Fig. 3, right). Through this link, we can interface our chip to virtual and real environments via address event communication, as well as configure neuron–synapse parameters via a shift-register scanner (not shown).

For testing, we focused on active power[1] and one-to-one equivalence with software. We also demonstrate that our chip can implement a well-known neural algorithm, a restricted Boltzmann machine (RBM), which acts as a front end to an off chip linear classifier for recognizing digits.

[1]Active power is our primary focus because passive power depends on the fabrication options, and can be addressed by process selection and availability.

A. Active Power

In our chip, active power scales linearly with spike activity since the design is purely event driven. To measure active power, we measure the *increase* in current beyond the baseline during high activity (averaged), where all neurons and axons are active in each time step (1kHz rate).[2] Our measurements are repeated over a range of supply voltages (Fig. 4); at $V_{dd} = 0.85$V, the core consumes just 45pJ/spike.

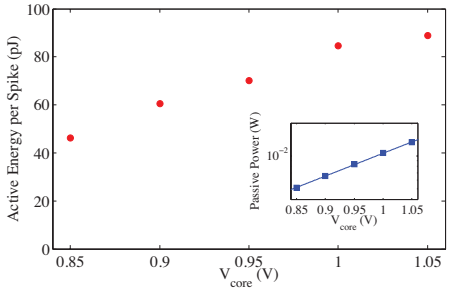


Fig. 4. Active energy per spike (red) decreases approximately linearly with lower V_{dd}, whereas the core's total passive power (blue, inset) decreases exponentially, shown on a log scale.

B. One-to-One Equivalence

To test that the chip satisfied one-to-one equivalence, we configured the synaptic strength and leak values of each neuron to `+1`, and the thresholds to `+100`. Then, we generated a pseudorandom connectivity where each synapse had a 20% probability of being 1. Lastly, the CPLD was set to route all neuron spikes back into the core (neuron 0,1,2 drove axon 0,1,2, respectively), creating a complex recurrent network.

Running the chip we observe that after 100 time steps, all the neurons spike in unison due to their identical positive leaks. This first barrage of spikes is routed back around to the axonal inputs, activating a pseudorandom pattern of excitatory recurrent connections; these inputs cause neurons to spike earlier in the next cycle, thereby having a de-synchronizing effect. Within a few cycles, the recurrently-driven activity dominates the dynamics leading to a complex spatiotemporal pattern. We simulated a software network with an identical configuration, and confirmed that the software and hardware have identical behavior (Fig. 5).

C. Implementing a Restricted Boltzmann Machine (RBM)

Our neurosynaptic core is capable of implementing a wide range of neural network algorithms, where weights are first learned offline, and then transformed into a hardware-compatible format. We present one example that implements a RBM, which is a well-known algorithm for classification and inference tasks. Specifically, we trained a two-layer RBM with 484 visible units and 256 hidden units on handwritten digits from the MNIST dataset. Our learning procedure followed directly from [5]; briefly, we use contrastive divergence to learn 484×256 real-valued weights to capture the probability

[2]Note that an axon event contributes less active power than a spike.

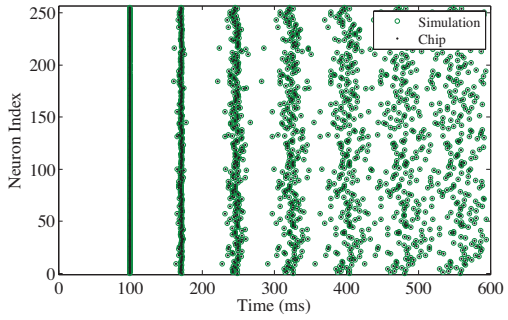


Fig. 5. The spiking dynamics of the chip exactly match a software simulation when configured with the same parameters and recurrent connectivity. Spikes are plotted as dots for measured chip data and circles for simulation.

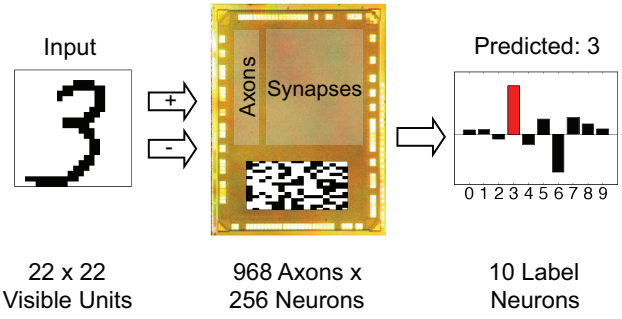


Fig. 6. (left) Pixels represent visible units, which drive spike activity on excitatory (+) and inhibitory (-) axons. (middle) 16×16 grid of neurons spike in response to the digit stimulus. Spikes are indicated as black squares, and encode the digit as a set of features. (right) An off-chip linear classifier trained on the features, and the resulting activation. Here, the classifier predicts that 3 is the most likely digit, whereas 6 is the least likely.

distribution of pixel correlations in the digits (60,000 images). After learning these weights, we trained 10 linear classifiers on the outputs of the hidden units using supervised learning. Finally, we test how well the network classifies digits on out-of-sample data (10,000 images), and achieved 94% accuracy.

To map the RBM onto our neurosynaptic core, we make the following choices: First, we represent the 256 hidden units with our integrate-and-fire neurons. Next, we represent each visible unit using two axons, one for positive (excitatory) connections and the other for negative (inhibitory) connections, accounting for 968 of 1024 axons. Then, we cast the 484×256 real-valued weight matrix into two 484×256 binary matrices, one representing the positive connections (taking the highest 15% of the positive weights), and the other representing the negative connections (taking the lowest 15% of the weights). Finally, the synaptic values and thresholds of each neuron are adjusted to normalize the sum total input in the real-valued case with the sum total input of the binary case.

Following the example from [6], we are able to implement the RBM using spiking neurons by imposing a global inhibitory rhythm that clocks network dynamics. In the first phase of the rhythm (no inhibition), hidden units accumulate synaptic inputs driven by the pixels, and spike when they detect a relevant feature; these spikes correspond to binary activity of a conventional (non-spiking) RBM in a single update. In the second phase of the rhythm, the strong inhibition resets all membrane potentials to 0. By sending the outputs of the hidden units to the same linear classifier as before (implemented off-chip) we achieve 89% accuracy for out-of-sample data (see Fig. 6 for one trial).

Our simple mapping from real-valued to binary weights shows that the performance of the RBM does not decrease significantly, and suggests that more sophisticated algorithms, such as deep Boltzmann machines, will also perform well in hardware despite binary weights.

V. Discussion

A long standing goal in the neuromorphic community is to create a compact, modular block that combines neurons, large synaptic fanout, and addressable inputs. Our breakthrough neurosynaptic core, with digital neurons, crossbar synapses, and address-events for communication, is the first of its kind to achieve this long standing goal in working silicon. The key new component of our design is the embedded crossbar array, which allows us to implement synaptic fanout without resorting to off-chip memory that can create an I–O bottleneck. By bypassing this critical bottleneck, it is now possible to build a large on-chip network of neurosynaptic cores, creating an ultra-low power neural fabric that can support a wide array of real-time applications that are one-to-one with software.

Looking forward, to build a human-scale system with 10^{14} synapses (distributed across many chips), our next focus is to tackle the formidable but tractable challenges of density, passive power, and active power for inter-core communication.

Acknowledgment

This research was sponsored by DARPA under contract No. HR0011-09-C-0002. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of DARPA or the U.S. Government. Authors thank William Risk and Sue Gilson for project management, Scott Fairbanks, Scott Hall, Kavita Prasad, Ben Hill, Ben Johnson, Rob Karmazin, Carlos Otero, and Jon Tse for physical design, Steven Esser and Greg Corrado for their work on digital neurons, and Andrew Cassidy and Rodrigo Alvarez for developing software for data collection.

References

[1] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. New York, NY, USA: ACM, 2009, pp. 63:1–63:12.

[2] C. Mead, *Analog VLSI and neural systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

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

[4] N. Imam and R. Manohar, "Address-event communication using token-ring mutual exclusion," in *Proceedings of the IEEE International Symposium on Asynchronous Circuits and Systems*, 2011.

[5] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006. [Online]. Available: http://dx.doi.org/10.1126/science.1127647

[6] P. Merolla, T. Ursell, and J. Arthur, "The thermodynamic temperature of a rhythmic spiking network," *CoRR*, vol. abs/1009.5473, 2010.