Editors: Volodymyr Kindratenko, kindr@ncsa.uiuc.edu
Pedro Trancoso, pedro@cs.ucy.ac.cy

MODELING SPIKING NEURAL NETWORKS ON SPINNAKER

By Xin Jin, Mikel Luján, Luis A. Plana, Sergio Davies, Steve Temple, and Steve B. Furber

SpiNNaker is a massively parallel architecture with more than a million processing cores that can model up to 1 billion spiking neurons in biological real time.

Despite an increasing amount of experimental data and deeper scientific understanding, deciphering the inner workings of biological brains remains a grand challenge. Investigations into the human brain's microscopic structure have shown that neuron cells are the key components in the cortex. Each individual neuron is physically very much like other cells in our body, but it's different in that it interacts with other neurons by receiving or sending electrical pulses, or *spikes*.

Researchers have proposed several mathematical models to describe the biological process of neurons firing spikes. These vary in their computational complexity as well as their fidelity, while maintaining biological plausibility. The spike is a common first class of abstraction among these various mathematical models. Spike events are communicated to all connected neurons, with typical fan-outs on the order of 10^3. Computational modeling of spiking neurons has abundant parallelism and no explicit requirement for cache coherent shared memory. Thus, researchers can use large supercomputing systems and high-performance computing clusters for this kind of simulation.[1] However, spike communication stresses standard HPC clusters and networks, making them unsuitable for real-time simulation.

In SpiNNaker, our treatment of spikes is a key innovation implemented with application-specific hardware: a multicast, packet-switched and self-timed communication fabric with on-chip routers. To maintain flexibility and generality, the neuronal models run in software on embedded ARM968 processors. These neuronal models communicate by means of *spike packets* directly supported by the SpiNNaker architecture.

We taped out the SpiNNaker test chips in 2009 with the batch arriving in Manchester in December. As Figure 1 shows, these test chips are fully functional SpiNNaker chips but have a highly reduced core count: only two cores per chip. Here, we offer an overview of our research project and describe the first experiments with these test chips running spiking neurons based on Eugene Izhikevich's model.[2] Note that we're not targeting artificial neural networks (such as perceptrons or multilayer networks) that were inspired by, but don't model, biologically plausible neural systems.

SpiNNaker Overview

SpiNNaker is a multicore-based architecture built for a specific purpose. The smallest configuration is a single SpiNNaker chip, which can simulate up to 20,000 neurons.[3] In this small form, the low-power properties of the ARM cores and asynchronous interconnect make SpiNNaker a feasible option for embedded control systems such as those in robots. This is a clear advantage over large HPC systems.

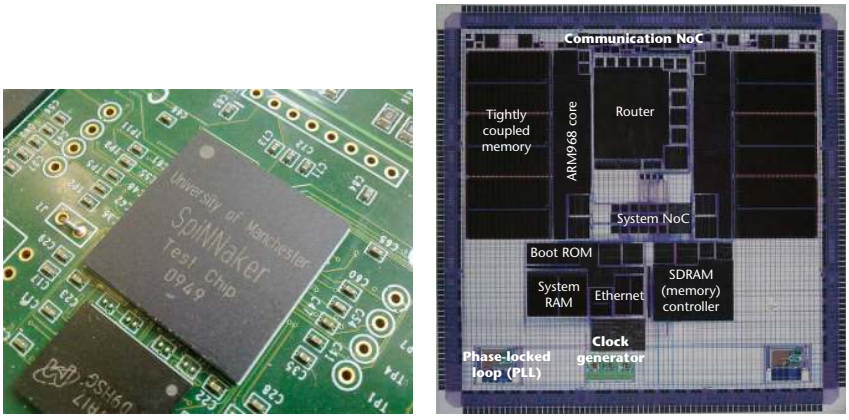
Compared to dedicated hardware solutions based on field-programmable gate arrays, analogue circuits, or hybrid analog-digital VLSI,[4] SpiNNaker offers flexibility in choosing neuronal dynamics, models, and learning rules. These are important features when we consider the experimental nature of state-of-the-art neural modeling.

The SpiNNaker chips are connected using a 2D toroidal triangular mesh based upon a light-weight packet-switched fabric.[5] Packets represent neural spikes and travel seamlessly through the fabric—a network-on-chip (NoC) and interchip connection network—that connects more than 65,000 (2^{16}) nodes housed in the largest SpiNNaker configuration. To emulate biological systems' very high connectivity, the on-chip routers provide multicast routing.

The SpiNNaker architecture is based on three guiding principles:

- *Virtualized topology.* The neural model's physical organization is decoupled from the target system's physical organization. So, in principle, any neuron can be modeled on any processing core in the system.

 Copublished by the IEEE CS and the AIP



(a) (b)

Figure 1. SpiNNaker test chip. (a) The test-board section showing a SpiNNaker chip and SDRAM. (b) A chip plot highlighting individual components.

Although the machine employs a 2D topology, it can model a 3D (or higher) neural structure. This is possible because electronic communication speeds are much higher than biological communication speeds.

- *Bounded asynchrony.* Time models itself. Because the system operates in real time (possibly scaled, although we assume ×1 scaling), there's no requirement for explicit synchronization in the computation. Things happen when they happen. Although this leads to nondeterministic behavior, the biological system being modeled shares this property.
- *Energy frugality.* Processors are free; the real cost of computing is energy. This is why we use embedded processors, and why the synchronous dynamic RAM (SDRAM) is of the mobile double data rate (DDR) variety—in both cases, we sacrifice some performance for greatly enhanced power efficiency.

As Figure 2 shows, each SpiNNaker chip has 18 identical ARM9 processing cores running at 200 MHz. As Figure 3 shows, the core is directly connected to two memory blocks: instruction tightly coupled memory (ITCM) and data tightly coupled memory (DTCM). The TCMs are small (32 and 64 Kbytes, respectively) but run extremely fast at processor clock speed; this is ideal for storing frequently accessed instructions or data.

One of the cores on each chip is selected to perform system management tasks. The other processing cores run independent event-driven neural processes, each simulating a group of neurons. The events are generated by peripherals, such as the direct memory access (DMA) control. Cores communicate with other cores and chips through the communication network. Access to all other on-chip resources on the system NoC is through the DMA controller, which is mainly used for reading (when a packet arrives) or writing (when updating synaptic information during learning) the neural state stored in the external SDRAM.

As Figure 2 shows, there's one router on each chip capable of on- and off-chip communication. It has 18 ports for internal use of the ARM cores and six ports to communicate with six adjacent chips. All ports are full duplex and implement self-timed protocols. The self-timed protocols make the SpiNNaker chip a globally asynchronous, locally synchronous design: individual processing cores on the chip act as (clocked) synchronous "islands" surrounded by a "sea" of asynchronous connectivity. This not only facilitates the VLSI design process, but isolates faulty cores and provides timing tolerance.

The router's internal organization is hierarchical; ports are merged in three stages before using the actual routing engine. The router is designed to support point-to-point and multicast communications. The multicast engine helps reduce pressure at the injection ports, and—compared to a pure point-to-point alternative—it reduces significantly the number of packets that traverse the communication fabric.

Packet routing must be done in an innovative way. Because neurons send spikes to thousands of other neurons, it's impractical to list all destinations in every packet. Therefore, routers make routing decisions based on the packet's source address (the identifier of the neuron that fired the spike). The network itself will deliver the packets to all chips containing neurons that have synaptic connections with the source neuron. These connections are embedded in the 1,024-word routing tables inside the routers, and must be preloaded using application-specific information. To minimize the space pressure on the routing tables, these offer a masked associative route look-up.

SpiNNaker chips are arranged in a 2D triangular torus topology with links to the neighbors in the north, south, east, west, southwest, and northeast. The routers perform a default routing that sends the packet following a straight line, a process that avoids using extra entries in the routing tables. For example, if the packet comes from the north, it will be sent to the south. The topology allows two-hop routes to go from a chip to each one of its neighbors. These two-hop paths between neighbor chips are known as *emergency routes* and the router can invoke them automatically to bypass problematic links due to transient congestion states or link failures.

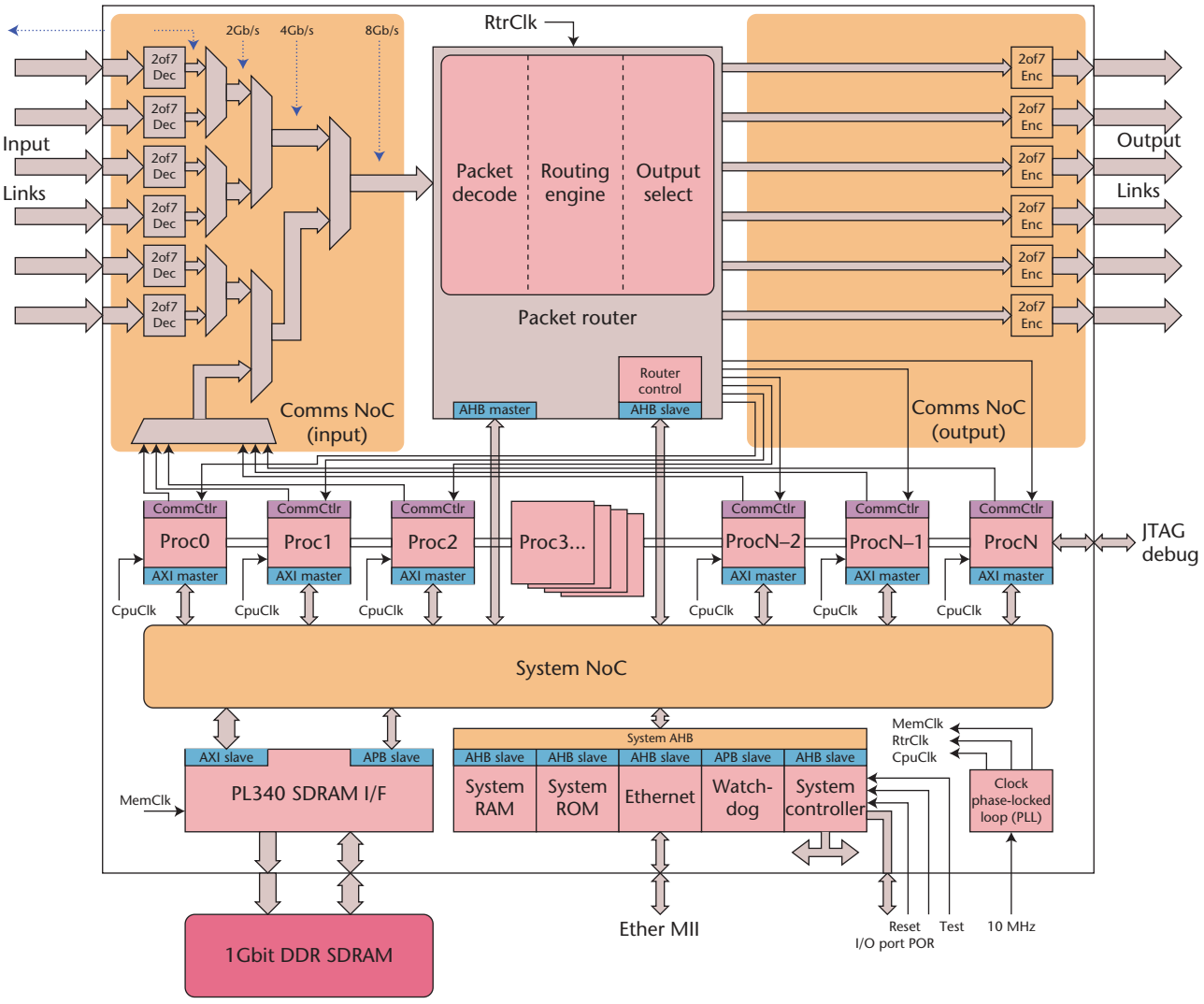


Figure 2. The SpiNNaker chip organization. The top half contains the multicast router that distributes spikes to cores on the same and neighboring chips. The ARM cores—used to simulate neurons—and their peripherals occupy the bottom half. The Joint Test Action Group (JTAG) is used for debugging. AHB stands for advanced high-performance bus and AXI stands for advanced extensible interface.

Neural Modeling: Implementing Izhikevich on SpiNNaker

Neuronal activity is a result of ionic movement, caused by two electrochemical gradients—concentration and electric potential gradients—around the cell body's membrane. The two electrochemical gradient forces drive ions in opposite directions, toward either the inside or the outside of the cell.

Several different types of mathematical models have been developed to describe neuronal dynamics. Alan Lloyd Hodgkin and Andrew Huxley developed a well-known biologically plausible conductance-based model (Hodgkin-Huxley) that describes ion currents dynamics by a set of nonlinear differential equations.[6] In conductance-based models, the variables and parameters have well-defined biological meanings, and can be measured experimentally. However, analyzing the conductance-based models is complicated.

In contrast, phenomenal models simply build relationships between the membrane potential (the electric potential difference between the inside and outside of the membrane) and input current. These models lack a direct biological meaning, but address most key properties of neurons and are less computationally intensive. One important phenomenal model is the Izhikevich model, which uses the *bifurcation theory* to reduce the high-dimensional conductance-based model to a 2D system with a fast membrane potential variable v and a slow membrane recovery variable u. The model's equations are

$$\dot{v} = 0.04v^2 + 5v + 140 - u + I,$$
$$\dot{u} = a(bv - u), \text{ and}$$
$$\textit{if } v \geq 30mv, \textit{ then } v = c, u = u + d,$$

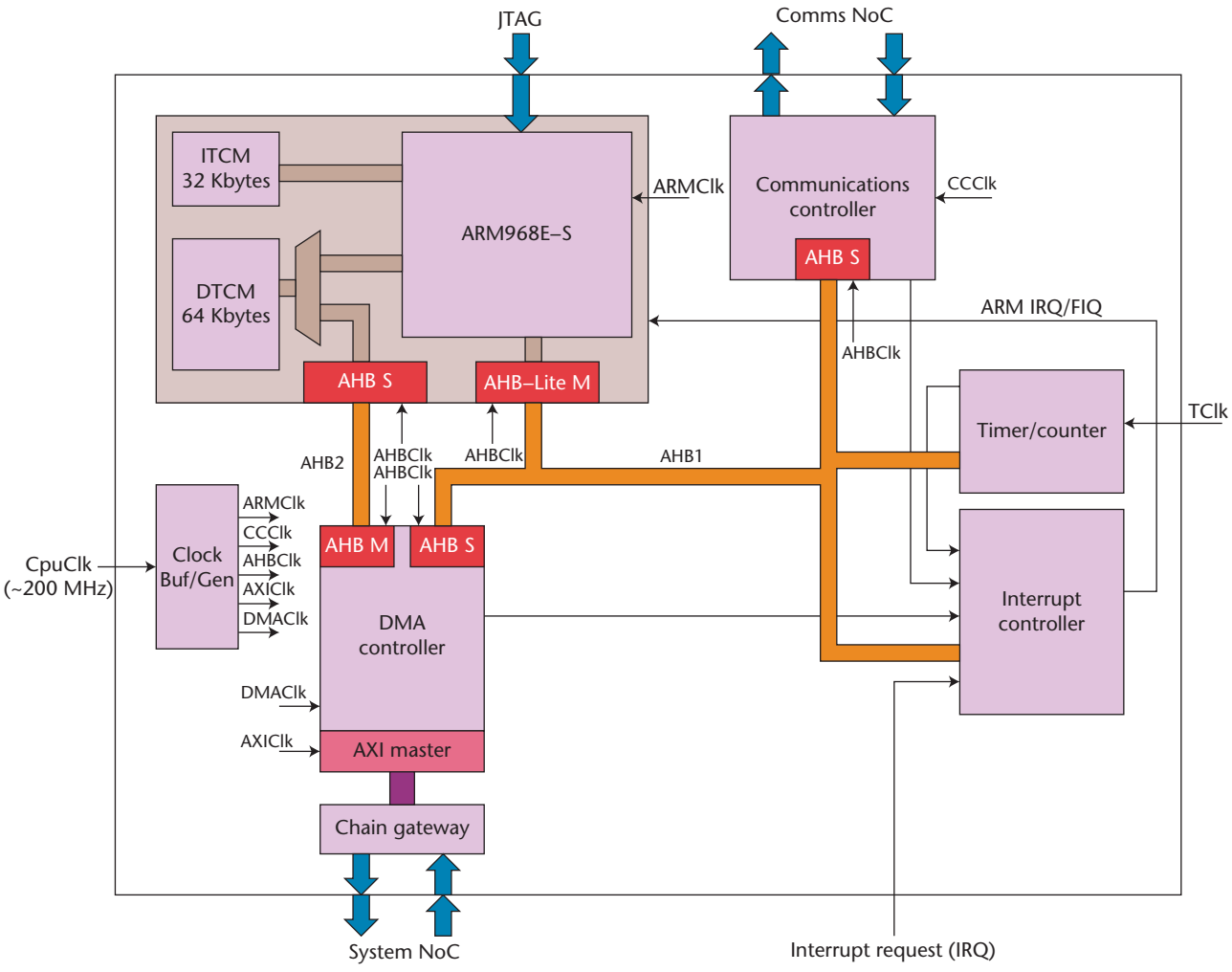


Figure 3. The processing core subsystem. Each ARM core has fast local memory to perform its tasks. The cores operate in interrupt-driven mode to respond in real time to system events such as spike arrivals. The core is directly connected to two memory blocks: instruction tightly coupled memory (ITCM) and data tightly coupled memory (DTCM). ARM stands for advanced RISC machines and IRQ/FIQ stands for interrupt request/fast interrupt request.

where $\dot{v} = dv/dt$, t is time in milliseconds, I is the synaptic current, v represents the membrane potential (in millivolts). The variable u represents the membrane recovery (also in mV), which reflects the negative effects on the membrane potential caused by factors such as the active K^+ and inactive Na^+ ionic current. Parameters a, b, c, and d are adjustable to reproduce the whole range of biological spiking firing patterns.

Accurate brain modeling requires not only the neuronal dynamics but also a comprehensive map of structural connection patterns in the human brain. Connectivity is closely related to the neural coding problem: information is coded and propagated within the neural network through structural links such as synapses and fiber pathways. Based on connectivity patterns discovered in laboratory experiments, researchers have built several mathematical models of connectivity.[7] The resulting connectivity knowledge was used to create large-scale neural network models, including Izhikevich's model to simulate brain activity.[8]

We selected the Izhikevich model to demonstrate real-time simulation with 1 ms resolution on SpiNNaker. We derived a 16-bit fixed-point arithmetic implementation to save both computation and storage space, as well as to avoid having a floating-point unit in the ARM968 cores. We take advantage of knowing the membrane potential's exact range of values ($-80 \leq v \leq 30$). By using a dual-scaling factor scheme, we can reduce the precision lost during the conversion and hence maintain a good precision level.[9] We also optimize the implementation's performance and accuracy by

- expanding the width from 16 to 32 bits during the computation to achieve better precision;
- transforming the equations to allow the use of efficient ARM instructions (`SMLAWB` and `SMLATT`,

which are signed multiply-accumulate operations);
- adjusting the parameters and precomputing Equation 1 as much as possible; and
- programming in ARM assembly language.

In our implementation, one iteration of Izhikevich equations takes six fixed-point arithmetic and two shift operations. This is more efficient than the original implementation, which requires 13 floating-point operations. In a practical implementation, the complete subroutine for computing Izhikevich equations can be performed in as few as 20 instructions if the neuron doesn't fire. If the neuron does fire, it takes 10 more instructions to reset the value and send a spike event. The detailed implementation, along with precision and performance analyses, is described elsewhere.[9]

To achieve low communication overhead, we use an event-address mapping (EAM) scheme. The EAM scheme keeps synaptic weights at the post-synaptic end (neurons receiving the spike) and set a relationship (in a mapping table) between the spike event and the address of the synaptic weight. Hence, no synaptic weight information needs to be carried in a spike packet. When a neuron fires, the packet propagates through a series of multicast routers according to the preloaded routing table in each router, and finally arrives at the destination processing cores.

The EAM scheme employs the two memory systems, DTCM and SDRAM, to store and access efficiently synaptic connections. DMA operations transfer each weight block from the SDRAM to the local DTCM, before computing the update following the Izhikevich equations. A core can easily find the synaptic weights associated with the fired neuron. It does this by matching the incoming spike packet with entries in the mapping table, which is organized as a binary tree.

Figure 4 shows a series of spike raster plots; the x-axis shows advancing biological simulation time, while the y-axis shows a simulated neuron. A point in a graph represents that the given neuron fired at the specified simulation time. The neural simulation involves a 500-neuron network with an excitatory-inhibitory ratio at 4:1. Each neuron is randomly connected to 25 other neurons. We randomly selected 12 excitatory and three inhibitory neurons as biased neurons, each receiving a constant input stimulus of 20 mV.

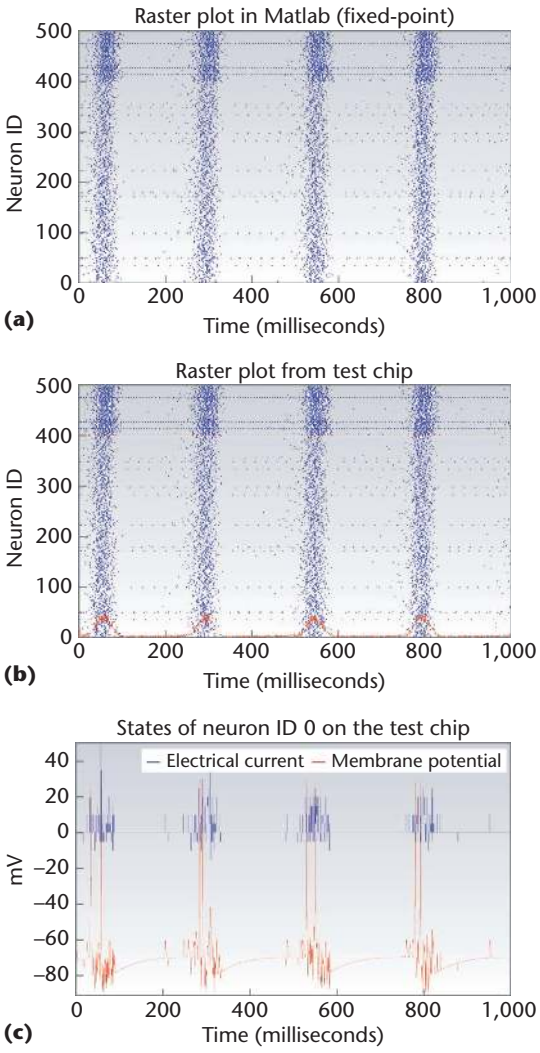
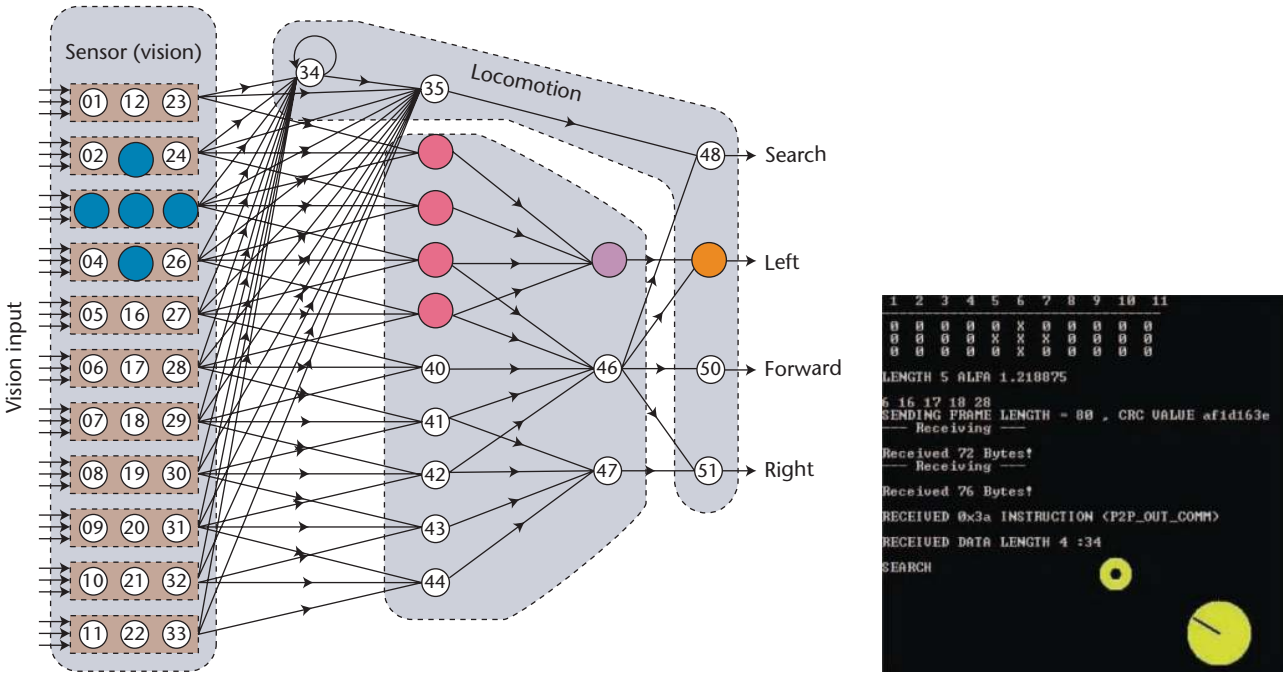


Figure 4. Spike raster plots of a 500-neuron model. (a) A 500-neuron, fixed-point Matlab simulation, (b) 500 neurons on a SpiNNaker test chip, and (c) the states of neuron 0 (excitatory) on a SpiNNaker test chip.

The spike raster plots compare a fixed-point arithmetic simulation in Matlab (Figure 4a) with the same neural model executing on a real SpiNNaker chip (Figure 4b). The spike timings match on both scenarios and show the same rhythm of 4 Hz. Figure 4c shows the activity of an excitatory neuron (ID 0) executing on the SpiNNaker test chip.

To test the SpiNNaker chips, we ported a doughnut hunter application. As Figure 5a shows, the hunter neuron network is composed of a few fast-spiking Izhikevich neurons. The application requires a server running on a host PC and a client running on SpiNNaker. The connection between the server and client is via an Ethernet interface.

The server models the environment with a doughnut and a hunter (see Figure 5b). The doughnut appears at a random location on the screen and the hunter moves to chase the doughnut. The server tracks the locations of the doughnut and the hunter, and provides the visual stimuli to the hunter. The SpiNNaker chip models the hunter's neural network: the server sends visual inputs, which are propagated to the motor neurons through the simple neural network.



(a) (b)

Figure 5. The doughnut hunter application running on a SpiNNaker Test Chip. (a) The doughnut hunter neural model. (b) The doughnut hunter in action.

The motor signals are sent back to the server to control the hunter movement.

These experiments demonstrate firmly, for the first time, that we have working silicon for SpiNNaker. However, we haven't yet demonstrated a real-time simulation of a billion (10^9) neurons—which is the objective of our research. To put that large number into perspective, a human brain contains approximately 10^{11} neurons. Having working silicon for SpiNNaker is a significant project milestone that takes us a step closer to our 2012 plan of deploying a SpiNNaker configuration with more than one million ARM cores.

Acknowledgments

We thank the Engineering and Physical Sciences Research Council (EPSRC), Silistix, and ARM for supporting this research. Mikel Luján is supported by a Royal Society University Research Fellowship. Arash Ahmadi at the University of Southampton developed the doughnut hunter application.

References

1. H. Markram, "The Blue Brain Project," *Nature Reviews Neuroscience*, vol. 7, 2006, pp. 153–160.
2. E.M. Izhikevich, "Simple Model of Spiking Neurons," *IEEE Trans. Neural Networks*, vol. 14, no. 6, 2003, pp. 1569–1572.
3. M.M. Khan et al., "SpiNNaker: Mapping Neural Networks onto a Massively Parallel Chip Multiprocessor," *Proc. Int'l J. Conf. Neural Networks*, IEEE Press, 2008, pp. 2849–2856.
4. P.A. Merolla et al., "Expandable Networks for Neuromorphic Chips," *IEEE Trans. Circuits and Systems*, vol. 54, no. 2, 2007, pp. 301–311.
5. L.A. Plana et al., "A GALS Infrastructure for a Massively Parallel Multiprocessor," *IEEE Design & Test*, vol. 24, no. 5, 2007, pp. 454–463.
6. A.L. Hodgkin and A. Huxley, "A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve," *J. Physiology*, vol. 117, no. 4, 1952, pp. 500–544.
7. O. Sporns, G. Tononi, and G. Edelman, "Theoretical Neuroanatomy: Relating Anatomical and Functional Connectivity in Graphs and Cortical Connection Matrices," *Cerebral Cortex*, vol. 10, no. 2, 2000, pp. 127–141.
8. E.M. Izhikevich, J.A. Gally, and G.M. Edelman, "Spike-Timing Dynamics of Neuronal Groups," *Cerebral Cortex*, vol. 14, no. 8, 2004, pp. 933–944.
9. X. Jin, S. Furber, and J. Woods, "Efficient Modeling of Spiking Neural Networks on a Scalable Chip Multiprocessor," *Proc. Int'l J. Conf. Neural Networks*, Ablex Publishing, 2008, pp. 2812–2819.

Xin Jin is an embedded software engineer in the Media Processing Division of ARM Ltd., and was involved in the SpiNNaker group during his doctoral studies. His research interests include neural modeling algorithms and software development on parallel neuromorphic hardware. Jin has a PhD in computer science from the University of Manchester. Contact him at jinxa@cs.man.ac.uk.

Mikel Luján is a Royal Society University Research Fellow in the School of Computer

Science at the University of Manchester. His research interests include managed runtime environments and virtualization, many-core architectures, and application-specific systems and optimizations. Luján has a PhD in computer science from the University of Manchester. Contact him at mikel.lujan@manchester.ac.uk.

Luis A. Plana is a research fellow in the School of Computer Science at the University of Manchester. His research interests include the design and implementation of systems-on-chip and on-chip interconnect. Plana has a PhD in computer science from Columbia University. He is a senior member of IEEE. Contact him at luis.plana@manchester.ac.uk.

Sergio Davies is a PhD student in the School of Computer Science at the University of Manchester. His research interests include artificial neural networks, neural network simulation, and synaptic plasticity. Davies has an MSc in telecommunication engineering from the University of Naples. Contact him at daviess@cs.man.ac.uk.

Steve Temple is a research fellow in the School of Computer Science at the University of Manchester. His research interests include VLSI design and hardware system design. Temple has a PhD in computer science from the University of Cambridge. Contact him at temples@cs.man.ac.uk.

Steve Furber is the ICL Processor of Computer Engineering in the School of Computer Science at the University of Manchester, where he leads the Advanced Processor Technologies research group. His research interests include multicore computing, energy-efficient VLSI design, and neural systems engineering. Furber has a PhD in aerodynamics from the University of Cambridge. He is a fellow of the Royal Society, the Royal Academy of Engineering, and IEEE, and is a 2010 Millennium Technology Prize Laureate. Contact him at steve.furber@manchester.ac.uk.

Selected articles and columns from IEEE Computer Society publications are also available for free at http://ComputingNow.computer.org.