JOURNAL OF THE ROYAL SOCIETY Interface

J. R. Soc. Interface (2007) **4**, 193–206
doi:10.1098/rsif.2006.0177
Published online 28 November 2006

REVIEW

Neural systems engineering

Steve Furber* and Steve Temple

School of Computer Science, University of Manchester, Oxford Road,
Manchester M13 9PL, UK

The quest to build an electronic computer based on the operational principles of biological brains has attracted attention over many years. The hope is that, by emulating the brain, it will be possible to capture some of its capabilities and thereby bridge the very large gulf that separates mankind from machines. At present, however, knowledge about the operational principles of the brain is far from complete, so attempts at emulation must employ a great deal of assumption and guesswork to fill the gaps in the experimental evidence. The sheer scale and complexity of the human brain still defies attempts to model it in its entirety at the neuronal level, but Moore's Law is closing this gap and machines with the potential to emulate the brain (so far as we can estimate the computing power required) are no more than a decade or so away. Do computer engineers have something to contribute, alongside neuroscientists, psychologists, mathematicians and others, to the understanding of brain and mind, which remains as one of the great frontiers of science?

Keywords: neural networks; computer engineering; computational neuroscience; massively parallel computing; spiking neurons

1. INTRODUCTION

Biological brains and engineered electronic computers fall into different categories. Both are examples of complex information processing systems, but beyond this point their differences outweigh their similarities. Brains are flexible, imprecise, error-prone and slow; computers are inflexible, precise, deterministic and fast. The sets of functions at which each excels are largely non-intersecting. They simply seem to be different types of system. Yet, throughout the (admittedly still rather short) history of computing, scientists and engineers have made attempts to cross-fertilize ideas from neurobiology into computing in order to build machines that operate in a manner more akin to the brain. Why is this?

Part of the answer is that brains display very high levels of concurrency and fault-tolerance in their operation, both of which are properties that we struggle to deliver in engineered systems. Understanding how the brain achieves these properties may help us discover ways to transfer them to our machines. In addition, despite their impressive ability to process numbers at ever-increasing rates, computers continue to be depressingly dumb, hard to use and totally lacking in empathy for their hapless users. If we could make interacting with a computer just a bit more like interacting with another person, life would be so much easier for so many people.

More fundamentally, understanding how the brain functions is one of the last great frontiers of science. 'Wet' neuroscience has revealed a great deal about the structure and operation of individual neurons, and medical instruments such as functional magnetic resonance imaging machines reveal a great deal about how neural activity in the various regions of the brain follows a sensory stimulus. But there is a wide gulf between these micro- and macro-level perspectives on brain function. Somewhere in this gulf are issues such as neural codes—how do populations of neurons jointly encode sensory and high-level information, modularity, temporal behaviour, memory (short- and long-term), attention and, of course, consciousness?

The objective of understanding the architecture of brain and mind is recognized as one of the grand challenges in computing research (Sloman 2004) and is a long-term multi-disciplinary project pursued at many different levels of abstraction.

The computer engineer brings a constructionist approach to these issues. Given the various different models that have been offered to describe the information processing function of an individual neuron, how do we go about selecting an appropriate model and constructing useful computational functions using it (instead of logic gates) as the basic component part? Can we build a library of such functions as a kit of parts from which to construct higher-level systems? Will the results of this enterprise deliver new and better ways to build computers, and/or will it tell us anything at all

*Author for correspondence (steve.furber@manchester.ac.uk).

Received 30 September 2006
Accepted 23 October 2006

about the biological systems that are the source of inspiration for this approach?

Therefore, we have two goals in this work: (i) to develop a 'neural toolkit' that can be used to build computers that share some of the properties of the brain, such as high levels of concurrency and fault-tolerance and (ii) to build a machine that will allow realistic simulation and study of the brain itself. In this review, we present a framework for this field of inquiry, suggest promising lines of attack and indicate when and where answers may emerge, so far as this can be foreseen.

1.1. The neuron

The basic biological control component is the neuron. A full understanding of the 'architecture of brain and mind' (Sloman 2004) must, ultimately, involve finding an explanation of the phenomenological observations that can be expressed in terms of the interactions between neurons.

Neurons appear to be very flexible components whose utility scales over systems covering a vast range of complexities. Very simple creatures find a small number of neurons useful. Honeybees find it economic to support brains comprising around 850 000 neurons, which give them exceptional navigational capabilities while travelling several miles from their hive. Humans have evolved to carry brains comprising 10^{11} neurons or so and use these to support exceptional motor control and complex societal interactions. Figure 1 illustrates a very small part of the cortex and gives an indication of the extent of the interconnections between neurons.

The basic logic gate used in digital circuits can be considered to be 'universal', in the sense that any digital circuit can be built using the same basic gate provided that a sufficient number of these gates are available—one structure can support all the functions within a class. The component neuron used across the range of biological brains is basically the same in its principles of operation, so in some sense it enjoys a universality similar to that of the logic gate in digital engineering, though the family of neurons employed in biological systems displays considerably more diversity in its members' physical characteristics.

There is a further similarity between neurons and logic gates: both are multiple-input single-output components. However, while the typical 'fan-in' (the number of inputs to a component) and 'fan-out' (the number of other components the output of a particular component connects to) of a logic gate are in the range 2–4, neurons typically have a fan-in and fan-out in the range 1000–10 000.

A more subtle difference between a logic gate and a neuron is in their internal dynamics. Whereas a logic gate implements a process that is essentially static and defined by Boolean logic, so that at any time (from a short time after the last input change) the output is a well-defined stable function of the inputs, a neuron has complex dynamics that includes several time constants, maintains a more complex internal state, and its output is a time-series of action potentials or 'spikes'. The information conveyed by the neuron's output is encoded in the timing of the spikes in a way that is not yet fully understood, although rate codes, population codes and firing-order codes all seem to offer valid interpretations.

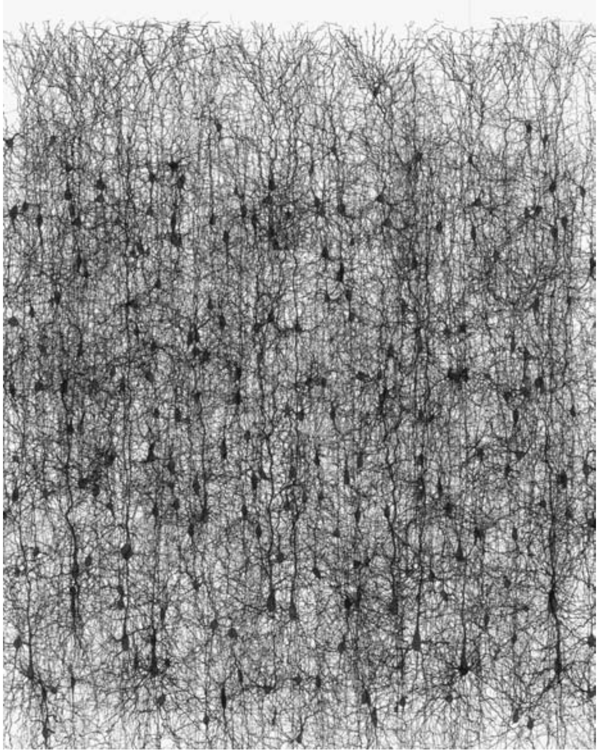

Figure 1. A view of the neuron cells and connections from a very small area of the cortex. Photo courtesy of the Brain Mind Institute, EPFL, Lausanne, Switzerland.

Accurate computer models of biological neurons exist, but they are very complex (e.g. Hodgkin & Huxley 1952). Various simpler models have been proposed that capture some of the features of the biology but omit others. The difficulty lies in determining which of the features are essential to the information processing functions of the neuron and which are artefacts resulting from the way the cell developed, its need to sustain itself and the complex evolutionary processes that led to its current form.

1.2. Neural microarchitecture

The universality of the neuron as a component is also reflected in certain higher-level structures of the brain. For example, the cortex displays a six-layer structure and a regularity of interconnect between the neurons in the various layers (Mountcastle 1978) that suggest the use here of a neural 'microarchitecture'. The same regular laminar cortical microarchitecture is in evidence across the cortex in regions implementing low-level vision processes such as edge detection and in regions involved in high-level functions such as speech and language processing. This apparent 'universality' (used here as defined earlier) of the cortical microarchitecture suggests that there are principles being applied, the understanding of which could offer a breakthrough in our understanding of brain function.

In contrast to the regularity and uniformity of the microarchitecture, the particular connectivity patterns that underpin these structures appear to be stochastic, guided by statistical principles rather than specific connectivity plans. The connectivity is also locally adaptive, so the system can be refined through tuning to improve its performance, a process termed 'neuroplasticity' (Schwartz & Begley 2003).

1.3. Engineering with neurons

As computer engineers we find the neuron's universality across wide ranges of biological complexity to be intriguing, and there is a real challenge in understanding how this component can be used to build useful information processing systems. There is an existence proof that this is indeed possible, but few pointers to how the resulting systems might work.

There are other 'engineering' aspects of biological neurons that are interesting too. We have already mentioned the regularity of neural microarchitecture. Neurons are physically much larger than transistors, having cell bodies with dimensions typically less than 30 μm, whereas today's transistors have dimensions below 0.1 μm. The power efficiency of neurons (measured as the energy required to perform a given computation) exceeds that of computer technology, possibly because the neuron itself is a relatively slow component. While computer engineers measure gate speeds in picoseconds, neurons have time constants measured in milliseconds. While computer engineers worry about speed-of-light limitations and the number of clock cycles it takes to get a signal across a chip, neurons communicate at a few metres per second. This very relaxed performance at the technology level is, of course, compensated by the very high levels of parallelism and connectivity of the biological system. Finally, neural systems display levels of fault-tolerance and adaptive learning that artificial systems have yet to approach.

1.4. Scoping the problem

The scale of the problem of modelling the human brain has been scoped by, among others, Mead (1990). The hundred billion neurons have of the order of 10^{15} connections, each coupling an action potential at a mean rate of not more than a few hertz. This amounts to a total computational rate of around 10^{16} complex operations per second. No computer has yet been built that can deliver this performance in real time, though this gap will be closed in the near future. Current supercomputer developments are aimed at delivering petaFLOP (10^{15} floating-point operations per second) performance levels, perhaps only one order of magnitude short of the performance required to model the human brain.

Perhaps more challenging is the issue of power efficiency. Moore (1965) observed that the number of transistors on a microchip doubled every year, and he predicted that this trend would continue for another 10 years. The doubling period was subsequently modified to 18 months, but has continued at this rate to this day and is expected to continue for at least another decade. The observation is widely referred to as 'Moore's Law'. In fact, Moore's Law ceased to be merely an observation a long time ago, becoming instead a semiconductor industry boardroom planning tool; therefore, it is now a self-fulfilling prophecy for as long as physics, engineering and industry economics allow. Although Moore's paper referred only to the increasing number of transistors on a microchip, the process of component miniaturization that makes this possible has also led to the spectacular improvements in performance, power efficiency and cost-per-function that underpins the pervasive digital technology that we enjoy today.

Mead argued that current computer technology will still be 10 million times less power efficient than biology even when Moore's Law has run its full course, so a real-time digital model of the brain will consume tens of megawatts of power. He argued that analogue electronics can close much of that gap, and he has built several silicon implementations of analogue neural systems that support this argument (Mead 1989). The very high power efficiency of the biological system has been emphasized more recently by Laughlin & Sejnowski (2003) and presents a real challenge to the computer engineer to find ways to build artificial systems that come anywhere close to matching it.

Delivering the necessary level of computational power is a pre-requisite to building a real-time model of the human brain, but it is far from being the only problem facing researchers in this area. One daunting challenge is the need to obtain the neural 'netlist', a term that we borrow here from electronic engineering where it is used to refer to a formalized description of an electronic circuit in terms of the type and parameters of each of the components (here neurons) and their connectivity patterns. A second challenge is to understand the developmental aspects of the brain's structure—the netlist is not static, but neurons grow and die and their interconnections extend and contract in response to activity and other biological factors. Thirdly, a brain needs a sensory system to provide its inputs and actuators to respond to its outputs—it needs to be embodied in some way in order to have a purpose for its activities. Fourthly, embodied brains do not live in isolation, they form societies and cultures that define and constrain their actions.

1.5. The research agenda

We use the term 'neural systems engineering' to describe the constructionist approach to exploring the potential of the neuron as a component in an information processing system (in the widest sense of this term).

This approach could be pursued entirely through software modelling; though because the computational demands of large-scale neural modelling are high, there have been many projects where special-purpose hardware has been constructed to accelerate the computation. Building the system in hardware also ensures that real-world issues such as noise are addressed. The special-purpose hardware may be aimed at modelling the low-level details of the neuronal processes in analogue electronic circuitry—an approach known as *neuromorphic* computing—or, at the other extreme, building massively parallel digital supercomputers with special features to support neural modelling.

The key issues at present are the following:

- To identify the simplest models that capture the information processing functions of a neuron. Neurons are very complex cells, but how much of this complexity is functionally relevant and how much is an artefact of the cell's evolutionary heritage, its need to grow, find energy, self-repair and so on?
- To represent the heterogeneous nature of biological neural systems. There are many different types of neurons in the brain, and each instance of a particular type has unique parameters.
- To identify and implement the necessary connectivity patterns of the natural system.
- To identify the neural 'codes' whereby populations of neurons represent complex information and through which the continuous sensory inputs can influence discrete actions and decisions.
- To identify the mechanisms of neural adaptation that enable the system to self-organize and learn, continually tuning and optimizing its performance.

Ultimately, it would be extremely useful to be able to raise the level of abstraction at which neural networks are modelled. If, for example, the functionality of the cortical microcolumn could be encapsulated in a set of mathematical equations, then the computational (and consequently power) demands of modelling the brain might come down by one or two orders of magnitude.

Among all these is the hope that some understanding will be gained of the emergent properties of complex dynamical systems, together with some insights into the fault-tolerant capabilities of biological systems and how these capabilities might be better emulated by engineered systems.

1.6. Structure of review

In §2, we look at the basic principles at work in neural computation. In §3, we look in detail at the problem we are addressing—what is known about the neuron's function and connectivity in its role as a component in complex biological systems—and we look at some of the models that are used to capture its function. In §4, we discuss the issues that arise in constructing large-scale artificial neural systems, and in §5, we look at how these issues have been addressed by various teams around the world, including our own work in this area. Section 6 concludes the paper with some speculation about the prospects for progress and potential breakthroughs in our understanding of brain function and our capability for engineering more intelligent systems in the future.

2. NEURAL COMPUTATION

In this section, we begin to look at the neuron as an information processing device. Any computational system must achieve a balance between its *processing*, *storage* and *communication* functions. It is useful to consider how these three functions are achieved in neural systems.

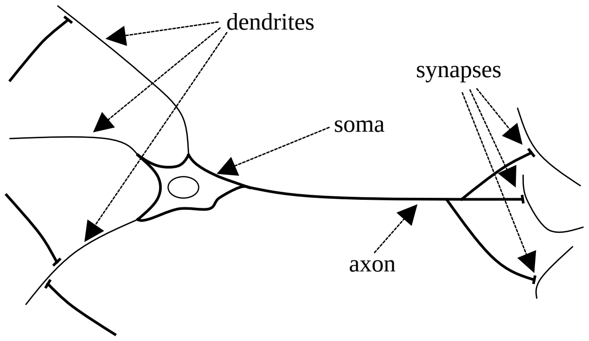

Figure 2. The four primary structures of a neuron. Inputs are collected via the dendrites and passed to the soma, the main body of the cell. The action potential (spike) generated in the soma propagates along the axon, where it passes through synapses to the dendrites of other neurons.

In discussing neurons, it is useful to use some biological terminology albeit, perhaps, with an engineer's interpretation of what this terminology signifies. A neuron may be viewed as comprising the following four structures (see figure 2):

- *Dendrites* are the tree-like structures that gather the inputs to the neuron from other neurons or sensory inputs and couple them to the soma.
- The *soma* is the central body of the neuron where the inputs are processed and the output is generated.
- The *axon* carries the output of the neuron through another tree-like structure to couple it to other neurons or physical actuators, incurring a signal-propagation delay that depends on the length of the axon.
- *Synapses* form the coupling between neurons. These can develop wherever the axon from one neuron is physically proximate to a dendrite of another. The coupling process incurs some time delay, but this can generally be added into the axonal delay for modelling purposes.

The synapse is the primary location of adaptation in the neural system: the strength of the coupling between two neurons self-adjusts over time in response to factors such as the correlation between the activities of the two neurons that are coupled through the synapse.

We can now look at how these structures contribute to the three aspects of computation that must be kept in balance: processing, communication and storage of information.

2.1. Processing

The processing function is performed within the neuron. The inputs are combined in the dendrites in some way and passed to the soma which produces output events in response to input events through a nonlinear transfer function, which we will model using suitable differential equations whose complexity is limited only by the available computing power. In some models, the dendrites simply sum the inputs, whereas in others they interact in more complex ways.

2.2. Communication

Communication in neural systems is predominantly through the propagation of spike 'events' from one neuron to the next. The output from the neuron's body—its soma—passes along its axon which conveys the spike to its many target synapses. Each synapse uses chemical processes to couple the spike to the input network—the dendritic tree—of another neuron.

Since the spike carries no information in its shape or size, the only information conveyed is in which neuron fired and when it fired.

2.3. Storage

It is in the storage of information that the neuron's story becomes most complex. There are many processes that can be seen as storing information, some operating over short time-scales and some very long term. Examples of these processess are as follows:

- the neural dynamics include multiple time constants, each of which serves to preserve input information for some period of time;
- the dynamical state of the network may preserve information for some time;
- the axons carry spikes at low speeds and therefore act as delay lines, storing information as it propagates for up to 20 ms; and
- the coupling strength of a synapse is, in many cases, adaptive, with different time constants applying to different synapses.

In a neural modelling system, we expect the model to capture the neural and network dynamics, and hence the contributions these mechanisms make to information storage. The axon delay-line storage does not come so easily as the high speeds of electronic signalling make spike communication effectively instantaneous. It is likely that the axon delay is functionally important, for example, enabling networks to learn complex spatio-temporal patterns as exhibited in polychronization (Izhikevich 2006), so we must put these delays back in, either by delaying the issue of the spike or by delaying its effect at the destination.

The primary long-term storage mechanism is synaptic modification (within which we include the growth of new synapses). This is the most fundamental storage mechanism; here, we require long-term stability and support for a range of adaptive mechanisms.

3. THE NEURON AS A COMPONENT

We take the neuron to be a device that, like a logic gate, has several inputs and a single output. The number of inputs will, however, typically be in the thousands rather than the two or three inputs that a logic gate normally has. The general scheme of the component neuron is illustrated in figure 3, which may be compared with figure 2 to see how it captures the major functional components of the biological neuron.

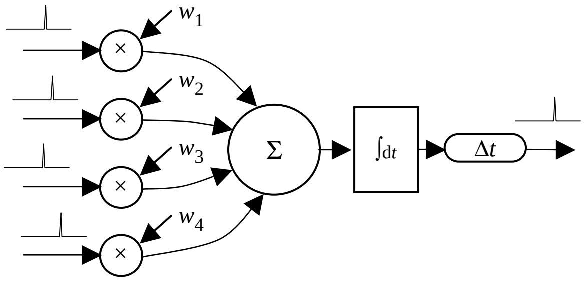

Figure 3. The neuron as a component. In the leaky integrate-and-fire model, input spikes are multiplied by their respective synaptic weights, summed and integrated over time. If the integral exceeds a threshold, the neuron fires and the integration restarts. The output spike may be delayed to model the propagation time along the axon.

3.1. Communicating with spikes

Although for most of its history the field of artificial neural networks has taken the output of a neuron to be a real value that varies from one discrete time-step to the next in a highly synchronous way, we will take the more biologically realistic model of the output as a time-series of action potentials (spikes) which, since the form of the spike is largely invariant, can be viewed as a time-series of asynchronous events. The output of neuron i is then simply

$$y_i = \sum_n \delta(t - t_{i,n}), \tag{3.1}$$

where $\delta(t)$ is the Dirac delta function representing a unit impulse at time t and $t_{i,n}$, $n=0, \ldots, N_i$, are the times of the spikes on neuron i.

In this model, information is conveyed solely in the times at which the events occur, perhaps in the rate of spiking, but there are other possibilities such as the relative timing of spikes from different neurons.

Although spikes appear to represent the primary means of propagating information in mammalian brains, they are clearly not the only means. Specialized neurons cause the emission of chemicals that have a broad effect on adjacent regions of neurons, and these are likely to be important in learning mechanisms. There is also evidence for direct analogue information exchange between neurons whose dendritic trees make contact.

We will proceed on the assumption that equation (3.1) captures the most important neural information exchange process, but remain aware that there may be other important processes in addition to spiking communication.

3.2. Point-neuron models

There are many different models that describe the internal operation of a neuron at different levels of detail. The simplest of these are point-neuron models, which ignore the spatial characteristics of the neuron. The inputs are combined through a weighted summing process to give a single driving force

$$I_i = \sum_j w_{ij} y_j, \tag{3.2}$$

where w_{ij} represents the strength of the synapse coupling neuron j into neuron i, and the sum is taken over all of the inputs y_j to neuron i.

This driving force is then applied to some form of nonlinear function to generate the output spikes from the neuron. A simple first-order differential equation is used in the *leaky integrate-and-fire* (LIF) model,

$$\dot{A}_i = I_i - A_i/\tau, \qquad (3.3)$$

$$\text{if } A_i \geq \vartheta \text{ fire neuron } i \text{ and reset } A_i = 0. \qquad (3.4)$$

Here, the activation, A_i, of neuron i decays to its zero rest state with time constant τ. It is increased by an amount w_{ij} every time input neuron j fires, and if at any time it exceeds the threshold ϑ, it fires and its activation is reset to zero.

The LIF model captures some of the essential behaviour of a biological neuron, but is often adapted to greater realism through the addition of features, such as

— *Habituation.* When presented with a step function in its input stimulus, a biological neuron tends to fire rapidly for a short period but does not sustain this firing rate for long, whereas the LIF model will fire at a steady high rate. Habituation can be added to the LIF model by making the threshold a leaky integrator of the neuron's own output,

$$\dot{\vartheta}_i = y_i - (\vartheta_i - \vartheta_0)/\tau_\vartheta, \qquad (3.5)$$

so each spike from the neuron increases its threshold above its rest state ϑ_0, but this effect decays with time constant τ_ϑ.

— *Refractory period.* Immediately after a neuron fires, it is insensitive to further inputs. The LIF neuron can be extended to model this in a number of ways, including simply causing it to ignore inputs for a fixed period after a spike, or resetting its activation after firing (equation (3.4)) to a negative level.

Adding these 'bells and whistles' to the LIF model increases both its biological accuracy and computational complexity. The trade-off between accuracy and computational complexity is a recurring theme in large-scale neural modelling.

3.3. The spike response model

The spike response model generalizes the LIF model and can describe the behaviour of any neuron that responds linearly to its inputs (Gerstner 1995). Each input causes a perturbation to the neuron's potential, which follows a characteristic course over time (which may depend on when this neuron last spiked). This can be captured by a kernel function $\varepsilon(t)$, and the activation potential of the neuron is then formed from a linear sum of these kernel functions, each scaled by its respective synaptic weight.

Similarly, when this neuron fires, its potential follows a characteristic time course that can be represented by another kernel function, $\eta(t)$,

$$A_i(t) = \eta(t - t_{i,N_i}) + \sum_j w_{ij} \sum_n \varepsilon(t - t_{j,n}). \qquad (3.6)$$

As with the LIF model, the neuron fires when the threshold is reached, and the threshold can be dynamic to model habituation. For computational efficiency, the kernel functions can be stored as look-up tables.

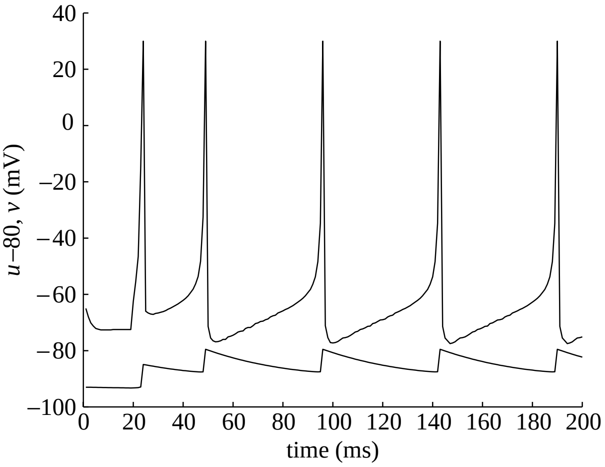

Figure 4. A solution to Izhikevich's equations, driven by an input step function at 20 ms. The slow variable, u (lower curve), has been offset by −80 mV for clarity. This solution was obtained using a 10 bit fixed-point approximation to the equations, but is very close to the real-valued solution.

3.4. The Izhikevich model

A rather different approach to the primary function of a point-neuron model is offered by Izhikevich (2004). His approach is to observe that the biological mechanism that gives rise to the neuron's spike output must have its basis in an instability in the electrochemical process that generates and sustains the spike as it propagates along the axon. He therefore turns to the mathematics of bifurcating processes to identify equations that capture the nature of this bifurcation at the lowest computational cost.

His investigations yielded the following pair of coupled differential equations:

$$\dot{v} = 0.04v^2 + 5v + 140 - u + I, \qquad (3.7)$$

$$\dot{u} = a(bv - u), \qquad (3.8)$$

$$\text{if } v \geq 30 \text{ then } v = c,\ u = u + d, \qquad (3.9)$$

where v is a 'fast' variable corresponding to the neuron's activation (A in equation (3.3), scaled to correspond to the activation of a biological neuron measured in millivolts); u is a 'slow' variable that adapts the neuron's dynamics over time; I is the weighted sum of inputs as in equation (3.2); and a, b, c and d are parameters that define the characteristic spiking patterns the neuron produces. The test for v reaching 30 mV in equation (3.9) is not a firing threshold, as was the test in equation (3.4), since equations (3.7) and (3.8) generate the spike directly. This test is detecting the peak of the spike itself.

An example of the behaviour of these equations is shown in figure 4, where the input I undergoes a step-function change at time 20 ms. The neuron spikes repeatedly, twice in quick succession and then at a slower, stable rate, displaying habituation. This figure was produced using a 10 bit fixed-point implementation of the equations, showing that relatively simple digital hardware is sufficient for this purpose.

The Izhikevich model is comparable in terms of computational complexity with the LIF model when the latter includes habituation and a refractory period, but it is able to model a much wider range of neural behaviours. As such, it is a very promising basis for large-scale neural modelling at the point-neuron level of complexity. Izhikevich has himself used it for simulating cortical activity with very large numbers of neurons (Izhikevich 2005).

3.5. Axons: the Hodgkin–Huxley model

Ground-breaking experiments on the giant axon of the squid (chosen because its size minimized the still-considerable experimental difficulties) culminated in the publication in 1952 of a seminal paper that presented equations describing the electrical behaviour of a nerve fibre (Hodgkin & Huxley 1952).

The equation for the axon membrane potential V is

$$C_{\mathrm{m}}\dot{V} = -g_{\mathrm{L}}(V-V_{\mathrm{L}}) - \bar{g}_{\mathrm{Na}}m^3h(V-V_{\mathrm{Na}}) - \bar{g}_{\mathrm{K}}n^4(V-V_{\mathrm{K}}), \tag{3.10}$$

where C_{m} is the membrane capacitance per unit area; g_{L}, $\bar{g}_{\mathrm{Na}}m^3h$ and $\bar{g}_{\mathrm{K}}n^4$are the conductances per unit area of the membrane for various ions that are involved in the axon processes; and V_{L}, V_{Na} and V_{K} are the associated equilibrium potentials. The dimensionless quantities m, h and n represent voltage-dependent channel variables which obey equations of the form

$$\dot{c} = \alpha(V)(1-c) - \beta(V)c, \tag{3.11}$$

where each channel variable has different $\alpha(V)$ and $\beta(V)$ functions that involve negative exponentials of the membrane voltage, V.

The Hodgkin–Huxley equations offer a very detailed model of the propagation of action potentials along neuronal fibres, but they are computationally demanding. Since the solution is characterized by a sharp transition between a continuous fluctuation for a weak input and a distinct action potential for a stronger input, spiking behaviour is often taken to be a good approximation, and the axon process is modelled as a simple delay, or possibly multiple delays, to allow for the different lengths of axon to different target synapses.

3.6. Dendritic trees and compartmental models

The dendritic networks that capture the inputs to a neuron are complex trees and may include nonlinear interactions between inputs on different branches. Point neurons generally ignore such effects and simply sum all the inputs. A more accurate model is to view the dendrites as similar to electrical cables and use transmission line models to compute the dynamics of input propagation towards the soma. As the dendritic trees are non-uniform and include branch points, 'compartmental' models split the trees into small cylindrical sections where each section has a uniform physical property (Bower & Beeman 1995). These models can be simple or highly detailed and can incorporate nonlinearities for greater accuracy.

As with the Hodgkin–Huxley model, great accuracy is achievable at the cost of considerable computational effort.

3.7. The synapse

The functional effect of a synapse is to transfer a spike from the axon at its input to the dendrite at its output. The biological synapse does this by releasing a number of packets of chemicals across the synaptic gap in response to the incoming spike, and these packets then affect the membrane properties of the receiving dendrite. The effect is quantized (Fatt & Katz 1952) and probabilistic in nature. It is often approximated by a multiplicative 'weight' as in equation (3.2).

Of much greater subtlety is the adaptive nature of the synapse. The ability of the synapse to adjust its effectiveness as a connection (in which we include the ability of the neuron to grow new connections) is believed to be the major long-term memory mechanism in the brain. Precisely how and when these changes take place is not fully understood, but there are a number of theories and explanations on offer.

Hebb (1949) postulated that when one neuron was close to another and repeatedly played a causal role in its firing, the coupling between them would strengthen. This postulate has since been validated experimentally, and the term 'Hebbian learning' is widely applied to mechanisms that modify the strength of a synapse as a result of correlations of various sorts between the spiking patterns of the two neurons that it connects. Long-term potentiation (LTP) is a term used to describe the most direct experimental confirmation of Hebb's principle (Lømo 2003). The opposite effect has also been observed in some areas of the brain—long-term depression (LTD), also known as 'anti-Hebbian learning', which describes a circumstance where correlations between two neurons result in a weakening of the synaptic strength.

Investigations into the detailed mechanisms of LTP and LTD have led to the observation of spike–time-dependent plasticity (STDP), where the scale of the synaptic modification has a well-defined dependency on the precise relative timing of the spikes from the input and output neurons, including changing sign if the order is not consistent with causality. Quite subtle models of the biophysical processes involved in STDP have been developed (Saudargiene *et al.* 2004).

STDP is unlikely to be the whole story, however. There are reward mechanisms in the brain that release chemicals that may modulate synaptic plasticity in some way, and mechanisms that lead to the growth of new connections (where causality cannot be involved, since before the growth there was no causal connection).

4. ENGINEERING NEURAL SYSTEMS

Now that we appreciate the behaviour of an individual neuron as a component and have a choice of models that we can use to represent its functionality, we can begin to consider the construction of systems of neurons. A number of questions arise in the development of any system of neurons, which reflect the issues listed previously in §1.5:

- At what level of detail should each neuron be modelled?

- — How do populations of neurons jointly encode information?
- — How is the connectivity of the network determined?
- — How is the connectivity of the network implemented?
- — How does the network learn, adapt or tune itself?
- — How is the network embodied and the body placed into an environment with which it interacts?

We will address each of these issues in the following sections, and then offer some examples of neural systems to illustrate how everything can come together.

4.1. Neural models

Much of the past work on building artificial neural networks has adopted the rate-coding view of neural output, where the only significant information that a neuron conveys about its inputs is in its firing rate (Adrian 1964). The firing rate can be represented by a real-valued variable, and the spiking behaviour of the biological neuron is abstracted entirely into this variable. Equation (3.2) is still used to compute the neuron's activation, but the variables y_j are real valued and no longer a time-series of spikes. The activation is modulated by a nonlinear transfer function to produce the real-valued output. The nonlinear transfer function is often a sigmoid function, which gives a smooth differentiable transition from 0 to 1 as the activation increases from its minimum to its maximum value.

In the most abstract models, the nonlinear transfer function is a simple threshold and all neural states are represented by binary values: 0 when the neuron is not firing and 1 when it is firing (McCulloch & Pitts 1943).

Recently, there has been a return to interest in more biologically accurate models that incorporate spiking behaviour. All of the models described in §3 have been explored, and others too. There is no consensus on the correct level of trade-off between complexity and biological realism, and none is likely to arise until many more questions have been answered about which features of the biological component are essential to its information processing function and which are simply artefacts of its biological origins.

4.2. Population encoding

The representation of sensory information by populations of neurons has been the subject of study. Eliasmith & Anderson (2003) offer detailed analytical tools for understanding how the combined spike firing rates of a heterogeneous population of neurons can represent a continuous physical parameter with arbitrary accuracy, and how successive layers of such populations can perform computations on those parameters.

An illustration of the representation of a continuously variable physical parameter is shown in figure 5. Here, a population of 50 LIF neurons, each with randomly selected sensitivity and offset (but all having the same maximum firing rate), together give an accurate estimate of the parameter even in the presence of noise. The error in the estimate is inversely proportional to the square root of the population size.

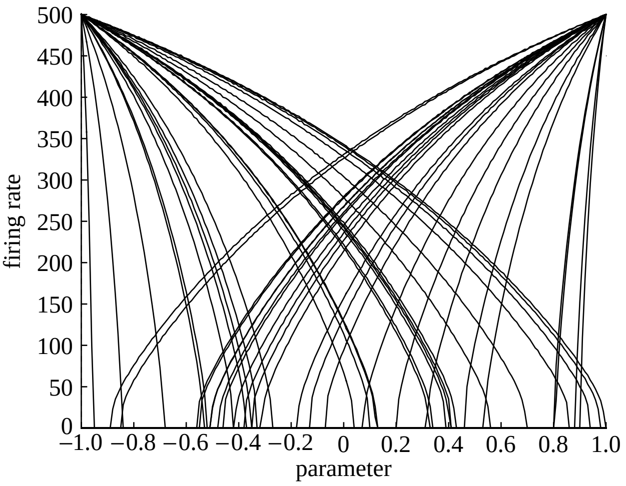

Figure 5. Representation of a physical parameter (x-axis) by the spike firing rates (y-axis) of a heterogeneous population of 50 neurons. Each curve shows the firing rate of one neuron (after Eliasmith & Anderson 2003).

In these population codes, each neuron is firing independently and the information is encoded across the individual firing rates of the neurons that form the population. This is not the only way in which a population of neurons can represent information, but the other encoding mechanisms described below all require that it is not only the firing rate of the neuron that is significant but also the timing of the individual spikes. Evidence that precise spike timing is important in biological systems is sparse, but one can point to O'Keefe & Recce's (1993) work on rat hippocampal place cells, where individual neurons fire in a well-defined phase relationship to the theta wave cycle.

Van Rullen & Thorpe (2001) take observations of the speed with which humans can respond to visual images, which does not allow enough time for any individual neuron to emit more than one spike, as evidence that individual firing rates are an insufficient explanation of the performance of the system. They have postulated *rank-order* codes as an alternative description. With rank-order codes, a population of neurons, in this case the ganglion cells in the retina, spike in an order determined by their tuning to the current sensory stimulus. The information is carried in the order of firing of the neurons in the population.

It is not necessary for the entire population to fire in a rank-order code, in which case information can be conveyed both in the order of firing of the subset that does fire and in the choice of that subset.

In a final simplifying step, it is possible to use just the subset choice to convey information and to ignore the order of firing altogether. In this case, the result is an N-of-M code, where all of the information is conveyed in the choice of the N neurons that fire from a total population of M neurons. Time has now been abstracted out of the model, and the system can be described and analysed in terms of pure binary patterns.

4.3. Spatio-temporal spike patterns

While unordered N-of-M codes are purely spatial and rely on approximate spike synchrony across the active subpopulation, rank-order codes represent a first step towards exploiting the temporal properties of spiking patterns to convey information. This can be taken further and generalized to polychronizaton (Izhikevich 2006). Here, the ability of an individual neuron to detect coincident inputs is combined with the intrinsic delays in the axons of the neurons that connect into its synapses to tune the neuron to respond to a very particular spatio-temporal pattern of activity on those input neurons. These spatio-temporal patterns can be very difficult to identify in neural firing traces, but have considerable information-bearing capacity and may play an important role in biological neural systems.

4.4. Defining connectivity

Abstract neural networks have a connectivity that can be defined algorithmically. Full connectivity is commonly used, where every neuron in one layer connects to every neuron in the next layer. In recursive networks, every neuron may connect to all of its peers in the same layer. Such networks have readily defined connectivity and are generally conceived in terms of this connectivity, which often has little to do with biological realism.

The availability of accurate, detailed network connectivity data for biological neural systems is fundamental to the task of building computer models of biologically realistic neural networks. Techniques for producing this connectivity information are improving, and recently Binzegger *et al.* (2004) have developed an approach based upon a statistical analysis of the proximity of the dendritic and axonal processes of different neuron populations in a three-dimensional reconstruction of the cat neocortex. This approach leads to highly detailed connectivity data that have been used to build computer models of large-scale biological neural systems (e.g. Izhikevich 2005).

4.5. Implementing connectivity

Biology employs massive numbers of low-speed channels to communicate neural spike events from their sources to their destinations. Despite the major advances in the density of microelectronic components, artificial neural systems cannot approach the physical connectivity of the natural systems.

However, electronic systems do have one advantage here: electronic communication is about five orders of magnitude faster than biology. An axon may carry tens to hundreds of spikes per second; a computer bus can operate at tens of MHz (Boahen 2000). Therefore, it is reasonable to seek ways to multiplex events from many neurons along the same bus or channel. A neural spike is an asynchronous event which carries information only in its timing, so the multiplexed information need simply identify the neuron that has fired. This led Sivilotti (1991) and Mahowald (1992) to propose the *address-event representation* of spikes, where each neuron in a system is given a unique number (address), and when the neuron fires this number is propagated through the interconnect to (at least) all neurons to which the firing neuron is connected.

A problem with multiplexing multiple asynchronous events through the same channel is that of collisions—when two events coincide closely in time, either one event must be dropped or they must be serialized, incurring some timing error. Boahen (2000) argues that asynchronous arbitration can be used to serialize the events with low timing error, even when 95% of the channel capacity is being used, and this approach scales well to faster technologies.

4.6. Learning, adapting and tuning

A key feature of any neural network, biological or engineered, is its ability to (i) learn new responses, (ii) adapt to new stimuli, and (iii) tune itself to improve its performance at the task in hand. These processes are generally achieved through the adjustment of synaptic weights in accordance with some sort of *learning rule*.

The long-standing way of optimizing artificial neural networks to a particular task is through the use of error back-propagation (Werbos 1994). Error back-propagation compares the outputs of a neural network with the desired output and then reduces the error in the output by adjusting weights and propagating errors backwards through the entire network. There are two problems with this approach for biological or large-scale engineered systems: (i) it assumes that the desired output state is known and (ii) it assumes the existence of an agent external to the system with global control of it. Neither of these is generally true for the systems of interest here. It is worth noting, however, that with suitable network topologies and local learning rules, biologically plausible systems have been shown to operate in ways that effectively deliver error back-propagation (O'Reilly 1996).

We will look for local learning rules that are based on Hebbian principles, adjusting weights according to local spike activity along the lines of STDP. In some cases this can be reduced to a simple rule, as in the case of the binary associative memories described in the §4.7. In more general applications this remains an uncertain aspect of the system engineering, where new insights and approaches are likely to lead to significant progress.

4.7. Example neural systems

Many applied neural systems are based on abstract neural models and do not depend directly on spike generation or communication. Examples that demonstrate principles of operation that could be employed in spiking systems with local learning rules include Willshaw *et al.*'s (1969) non-holographic memory and its close relative, the correlation matrix memory (CMM; Kohonen 1972). CMMs employ binary output neurons with binary synaptic weights and often use N-of-M population codes. They have proved very effective for building large-scale associative search systems such as the AURA system at the University of York (Austin *et al.* 1995), which has found a wide range of industrial applications.

Similar abstract models are employed in sparse distributed memories (SDMs). Although Kanerva's (1988) original SDM was not directly compatible with standard spiking neuron models, our variants of Kanerva's SDM employing N-of-M codes (Furber *et al.* 2004) or rank-order codes (Furber *et al.* in press) can readily be implemented with such models.

4.8. Neuromorphic systems

An approach to engineering artificial neural systems that have been explored in certain application domains is to implement neural models in analogue hardware. Some functions such as multiplication are very much cheaper and less power hungry when implemented in analogue rather than digital electronics, and analogue systems also offer intriguing nonlinearities that offer elegant solutions to certain tricky aspects of neural modelling.

The analogue approach has been applied to vision systems (Mead 1989; Lichtsteiner *et al.* 2006; Yang *et al.* 2006) and similar early-stage sensory input processing. The combination of analogue neural models with digital spike communications models the biological solution closely and is probably the most promising microelectronic approach to building large-scale neural networks in the long term. In the shorter term, the uncertainties over the optimal neural model make the inflexibility of the analogue implementation (compared to a programmable digital system) unattractive for the general-purpose neural processor.

5. LARGE-SCALE PROJECTS

It seems reasonable to assume that some of the most challenging and interesting aspects of neural function will be manifest only in systems of considerable scale where, for example, there is scope for very high-dimensional representation and processing of sensory input. This makes the construction of large-scale systems an important component of neural systems engineering research. There have been several projects in recent times aimed at building large-scale computer models of neural systems, using different complexities of model and different techniques to accelerate the computation:

— Software systems run on conventional machines are highly flexible but can be slow, unless run on very powerful computers such as Blue Brain (Markram 2006). Izhikevich (2005) ran a simulation of one second of activity in a thalamocortical model comprising 10^{11} neurons and 10^{15} synapses, a scale comparable with the human brain. The simulation took 50 days on a 27-processor Beowulf cluster machine.

— Field-programmable gate arrays (FPGAs) have the potential to accelerate key software routines (Zhu & Sutton 2003), though it can be difficult to get the correct system balance between processing and memory. FPGAs are very flexible but somewhat harder to program than software, and high-performance computer manufacturers are very interested in the possibility of integrating FPGAs into their parallel machines. This may result in software tools that deliver the acceleration transparently to the programmer.

— Building bespoke hardware to support neural modelling is an approach that has been tried from time to time with limited lasting success. The fundamental problem is the same as for other areas of application-specific hardware—the commercial momentum behind the progress of the general-purpose computer renders any benefit of special-purpose hardware hard-won and short-lived. In the neural modelling area there is also the issue of deciding how much the neural model should be cast into hardware, optimizing performance but losing flexibility, against making the system as soft and general-purpose as possible.

Here, we give brief descriptions of three projects aimed at large-scale neural modelling: Blue Brain, at EPFL, Switzerland; SPINN, at the Technical University of Berlin; and our own plans for the SpiNNaker machine.

5.1. Blue Brain

By far, the largest-scale project aimed at building computer simulations of sections of the brain is the Blue Brain project at EPFL in Switzerland (Markram 2006). This work is based upon one of the world's most powerful supercomputers, the IBM Blue Gene/L. This machine delivers up to 360 teraFLOPS of computing power from 8192 PowerPC CPUs each running at 700 MHz and arranged in a toroidal mesh. Alongside the IBM supercomputer is a sophisticated stereo visualization system based upon SGI graphics computers.

The Blue Brain project simulates biological neural networks using detailed compartmental neuron models and aims to deliver biologically accurate models of neural microcircuits such as the neocortical microcolumn. The computations are based upon the Hodgkin & Huxley (1952) equations and Rall's (1959) cable models of the dendritic and axonal trees and use the NEURON (2005) simulator codes, extended to use a message-passing interface to communicate action potentials between neurons modelled on different processors.

In addition to exploiting their computational resources, the Blue Brain team is also assembling a major database of biological neural data upon which to base their computer models.

The Blue Brain project as currently configured has a machine capable of simulating up to 100 000 very complex neurons or 100 million simple neurons. The emphasis is on maximally accurate models of biological neural systems.

5.2. SPINN

The Spiking Neural Network activity at the Technical University of Berlin has yielded a series of hardware systems for the acceleration of neural network modelling over more than a decade of related research projects: BIONIC, NESPINN, MASPINN, SP^{2}INN and, most recently, SPINN Emulation Engine (SEE).

The Biological Neuron IC (BIONIC) project yielded a chip capable of modelling up to 16 neurons each with 16 synapses (Prange & Klar 1993). The NESPINN (Neurocomputer for Spiking Neural Networks) project aimed to model 16 000 neurons each with 83 synapses (Jahnke *et al.* 1996), with the possibility of handling larger systems with multiple accelerator boards.

The Memory Optimized Accelerator for Spiking Neural Networks (MASPINN; Schoenauer *et al.* 1998) was a hardware accelerator that connects to a host PC via a standard PCI bus. The neural model treated the dendritic tree as a set of independent leaky integrators, each receiving a number of inputs. The outputs of these integrators then interact in a programmable way to form the driving current for a soma model with a dynamic threshold (again generated by a leaky integrator). Axonal delays are modelled at the neuron's output, so a neuron that connects with different delays to other neurons has multiple outputs, one for each delay value. MASPINN employs a number of optimizations to reduce the computational demands. It caches synaptic weights that are used heavily, and it tags inactive components so that they do not consume resource computing the leaky integrator function. The MASPINN project aimed to simulate a million relatively simple neurons and had a specific application area—image processing—as its target.

The Synaptic Plasticity in Spiking Neural Networks (SP^{2}INN) project (Mehrtash *et al.* 2003) aimed at building hardware to model a million neurons with several million synaptic connections but, at the end of the paper, the authors contemplate the difficulties of designing special-purpose hardware to compete with the relentless advances in the performance of general-purpose computers.

The SEE project abandons custom hardware in favour of FPGAs and exploits the embedded general-purpose processing power incorporated in some of today's FPGA architectures (Hellmich *et al.* 2005). The system can model half a million neurons each with 1500 synaptic connections.

Taken together, these projects represent a considerable body of experience in designing hardware to support spiking neural network modelling, and it is instructive to see how each project has built on the ideas displayed in its predecessors but how little the hardware components, presumably designed with considerable effort, have carried forward. This illustrates the difficulty of making bespoke hardware flexible enough to solve more than the problem of the moment.

5.3. SpiNNaker

The SpiNNaker project at the University of Manchester (Furber *et al.* 2006*b*) has as its goal the development of a massively parallel computer based on chip multiprocessor technology and a self-timed Network-on-Chip (NoC) communications system (Bainbridge & Furber 2002). The system is aimed at modelling large-scale systems of up to a billion spiking neurons in real time and is optimized for point neurons such as the LIF and Izhikevich models. It is not intended to run models with high biological accuracy, but is much more aimed at exploring the potential of the spiking neuron as a component from which useful systems may be engineered. Biological data are taken as a very useful source of inspiration, but not as a constraint, and useful ideas for novel computation systems will be seen as a positive outcome irrespective of their biological relevance.

The philosophy behind the system architecture is based on the observation that modelling large systems of spiking neurons falls into the 'embarrassingly parallel' class of applications, where the problem can be split into as many independent processing tasks as is useful. The performance of an individual processor in the system is not an important parameter. What matters is the cost effectiveness of the implementation, which can be broken down into the capital cost and the running cost, which can be assessed, respectively, in terms of:

— MIPS (millions of instructions per second) per mm^2: how much processing power can we get on a given area of silicon? And,

— MIPS per watt: how energy efficiently can this processing power be delivered?

The choice for this system is between employing a small number of high-end processors or a larger number of lower-performance embedded processors on each chip. The performance density (MIPS per mm^2) of both classes of microprocessor is similar, but the embedded processors are an order of magnitude more energy efficient. Hence, the decision is to use embedded processors in large numbers, and the SpiNNaker chip will incorporate up to 20 ARM processor cores to execute the neural modelling code.

The organization of the SpiNNaker multiprocessor chip is illustrated in figure 6. One of the processors on the chip is selected to act as monitor processor, and runs the operating system functions on the chip. The remaining processors act as fascicle processors, each modelling a group of up to a thousand individual neurons where that group is selected to have as much commonality as possible in the neurons that connect into its member neurons and the neurons elsewhere that its member neurons connect to.

Each fascicle processor receives spike events from, and issues spike events into, a packet-switching communications system, with each spike event encoded as a single packet. Within a chip, these spike events converge through the NoC Communications to an arbiter, where they are selected and sent in sequence to a router (Furber *et al.* 2006*a*). The router uses internal tables to identify which fascicle processors should receive each event (which can be determined from the connectivity netlist of the neural network that is being modelled) and passes the event on accordingly; this mapping may include none, one or several of the fascicle processors, so a full multicast routing mechanism is required that is based on address-event communication as described in §4.5.

The Communications NoC also extends between chips, so the system can be enlarged by connecting multiple chips together, as illustrated in figure 7. Each chip has six transmit interfaces ('Tx i/f' in figure 6) and six receive interfaces ('Rx i/f') that effectively extend the Communications NoC to six neighbouring chips through bi-directional links. The Router can direct packets from any on- or off-chip source to any on- or off-chip destination and has sufficient capacity to support systems comprising very large numbers (up to tens of thousands) of chips.

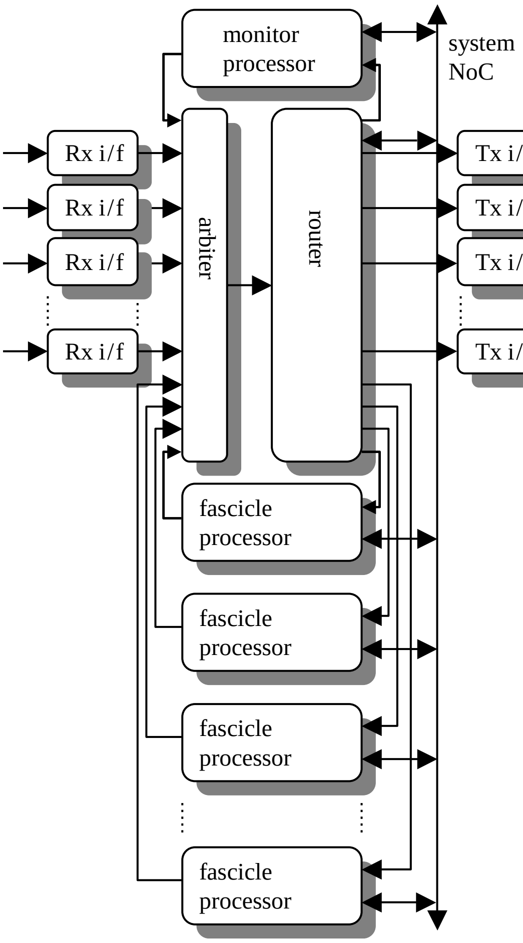

Figure 6. Organization of a SpiNNaker chip multiprocessor node, illustrating the Communications Network-on-Chip (NoC) that is used to carry spike event packets around the system. Each fascicle processors models many neurons. Packets from other nodes arrive through the receiver interfaces ('Rx i/f') and are merged with packets issued by the fascicle processors into a sequential stream by the arbiter. Each packet is then routed to one or several destinations, which may include other processing nodes (via the transmit interfaces—'Tx i/f') and/or local fascicle processors. The monitor processor carries out operating system functions and provides visibility to the user of on-chip activity.

5.4. Virtual communications

The SpiNNaker architecture illustrates one of the important principles of engineering large-scale neural modelling systems: the need to decouple the physical organization of the engineered system from the physical organization of the biological system it is designed to model.

Biological neural systems develop in three dimensions, though the way the three-dimensional space is used is quite variable. The cortex, for example, is often described as a thin sheet of neurons, where the third dimension is used largely for long-range connections and to enable the large two-dimensional area to be folded into a convoluted shape in order to fit into a small three-dimensional volume. On a small scale, the sheet does have a thickness which is divided into a characteristic six-layer structure.

The SpiNNaker system architecture, as illustrated in figure 7, has a strongly two-dimensional structure. However, this does not imply in any way that it can only model two-dimensional neural structures. Indeed, SpiNNaker can model neural networks that are formed in two, three or even more dimensions. The key to this flexibility is to map each neuron into a virtual address space, which means that each neuron is assigned a unique number. The assignment can be arbitrary, though an assignment related to physical structure is likely to improve the modelling efficiency. Then neurons are allocated to processors; again in principle the allocation can be arbitrary, but a well-chosen allocation will lead to improved efficiency. Finally, the routing tables must be configured to send spike events from each neuron to all of the neurons to which it connects, and this can be achieved using the neurons' addresses.

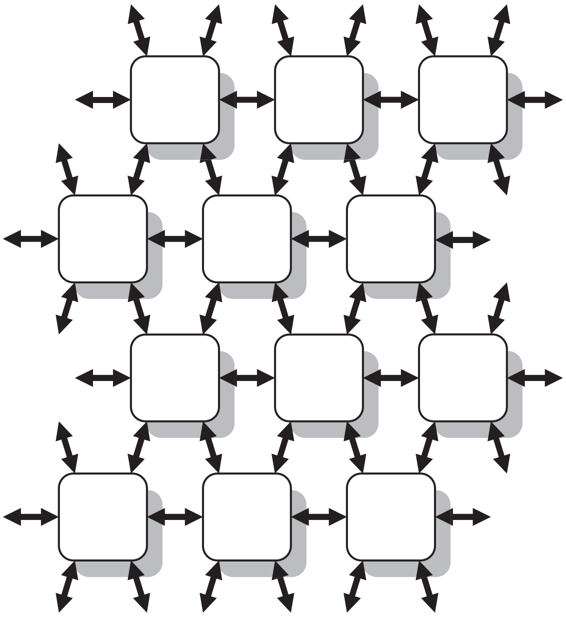

Figure 7. SpiNNaker system architecture. Each of the chip multiprocessor nodes is connected to its six nearest neighbours by bi-directional links. The left and right sides of the mesh are connected, as are the top and bottom edges, to form a two-dimensional toroidal surface.

The dissociation between the physical organization of the computer system and the physical organization of the biological system it is being used to model is possible owing to the very high speed of electronic communications relative to the speed of propagation of biological signals. This means that the delays inherent in getting a spike event across many chips in the SpiNNaker system are negligible on the time-scales of neuronal processes. There is a drawback to the high speed of electronics, however. The physical delays in the biological system are likely to be functionally important; therefore, they must be re-instated in the electronic model. This is one of the more difficult and expensive aspects of the computational task.

5.5. Diverse approaches

Blue Brain and SpiNNaker are both highly parallel systems employing large numbers of general-purpose processors to deliver flexibility (through programmability) in the neuron models that they support. Beyond this apparent similarity, however, there are marked differences in the way these two machines will be used to investigate neural computation.

The Blue Brain project emphasizes biological fidelity and as a result uses high-performance processors with support for high-precision real-valued arithmetic which allows complex equations to be solved efficiently. The SpiNNaker design emphasizes the real-time modelling of very large numbers of much simpler equations; therefore, it uses simpler processors which support only integer arithmetic, which can still yield accurate solutions to differential equations as illustrated in figure 4, though considerable care must be taken over operand scaling.

There are many other differences between the two machines, with Blue Brain using an ‘off-the-shelf’ supercomputer while SpiNNaker is a bespoke design, the latter therefore having a lightweight communications architecture highly tuned to the neural modelling application.

While there remains so much uncertainty about the fundamental principles of biological neural processing, the diversity of approach reflected in the differences between Blue Brain and SpiNNaker (and neuromorphic and FPGA-based systems) is to be welcomed. No one knows which of these approaches is the most promising. It is our belief, based upon our experience as computer engineers, that large-scale complex systems are unlikely to be robust if they depend critically on the fine detail of the components from which they are constructed, so we are looking for explanations and insights at the network level and ignoring much of the biological detail. Whether or not this belief is justified only time will tell, but this is the belief that is driving the current direction of the SpiNNaker project.

6. FUTURE PROSPECTS

A great deal is known about the function and behaviour of the neuron, but a great deal more remains to be revealed. If estimates of the computing power required to model a neural system of the complexity of the human brain are not grossly misconceived, computers fast enough to do the job are not far away. But computing power alone will not solve the problem.

It is not predictable when or where a breakthrough in our understanding of brain function will emerge, so there is considerable merit in the diversity of approaches that are now in evidence. The core activities will remain for some time the painstaking bottom-up laboratory work of the neuroscientist and the top-down human-centric approach of the psychologist. But alongside these, there are challenges for computational neuroscientists, computer scientists, and electronic and computer engineers, all of whom can find opportunities to explore the potential of the neuron as inspiration for novel ideas in computation.

There are many possible approaches within the constructionist territory, some of which we have indicated in this paper. The spectacularly well-resourced Blue Brain project has the computing power to build highly accurate models of biological systems, and we can expect dramatic insights into the operation of complex neural systems to arise from that work, to complement the exotic images and visualization facilities they have already demonstrated. With our own work, we are leaving a lot of the biological complexity behind and working with more abstract neural models, with the expectation that the world of complex event-driven dynamical systems will yield insights both into the biology that we employ loosely as inspiration for our work and into novel models of computation.

S.T. is supported by the EPSRC Advanced Processor Technologies Portfolio Partnership at the University of Manchester. S.F. holds a Royal Society-Wolfson Research Merit Award. The SpiNNaker research is supported by EPSRC, and by ARM Ltd and Silistix Ltd. The support of these sponsors and industrial partners is gratefully acknowledged.

The authors would also like to acknowledge the constructive suggestions for improvement offered by the anonymous referees who reviewed this paper prior to its publication.

REFERENCES

Adrian, E. D. 1964 *Basis of sensation*. London, UK: Haffner Publishing Company.

Austin, J., Kennedy, J. & Lees, K. 1995 The Advanced Uncertain Reasoning Architecture, AURA. In *Proc. WNNW'95*.

Bainbridge, W. J. & Furber, S. B. 2002 CHAIN: a delay-insensitive chip area interconnect. *IEEE Micro.* **22**, 16–23. (doi:10.1109/MM.2002.1044296)

Binzegger, T., Douglas, R. J. & Martin, K. A. C. 2004 A quantitative map of the circuit of cat primary visual cortex. *J. Neurosci.* **24**(39), 8441–8453. (doi:10.1523/JNEUROSCI.1400-04.2004)

Boahen, K. A. 2000 Point-to-point connectivity between neuromorphic chips using address events. *IEEE Trans. Circuits Syst.* **47**(5), 416–434. (doi:10.1109/82.842110)

Bower, J. M. & Beeman, D. 1995 *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*. New York, NY: Springer.

Eliasmith, C. & Anderson, C. H. 2003 *Neural engineering*. Cambridge, MA: MIT Press.

Fatt, P. & Katz, B. 1952 Spontaneous subthreshold activity at motor nerve endings. *J. Physiol.* **117**, 109–128.

Furber, S. B., Bainbridge, W. J., Cumpstey, J. M. & Temple, S. 2004 A sparse distributed memory based upon N-of-M codes. *Neural Netw.* **17**(10), 1437–1451. (doi:10.1016/j.neunet.2004.07.003)

Furber, S. B., Temple, S. & Brown, A. D. 2006*a* On-chip and inter-chip networks for modelling large-scale neural systems. In *Proc. ISCAS'06*, pp. 1945–1948.

Furber, S. B., Temple, S. & Brown, A. D. 2006*b* High-performance computing for systems of spiking neurons. In *Proc. AISB'06 workshop on GC5: architecture of brain and mind*, vol. 2, pp. 29–36.

Furber, S. B., Brown, G., Bose, J., Cumpstey, M. J., Marshall, P. & Shapiro, J. L. In press. Sparse distributed memory using rank-order neural codes. *IEEE Trans. Neural Netw.*

Gerstner, W. 1995 Time structure of the activity in neural network models. *Phys. Rev. E* **51**, 738–758. (doi:10.1103/PhysRevE.51.738)

Hebb, D. O. 1949 *The organization of behavior: a neuropsychological theory*. New York, NY: Wiley.

Hellmich, H. H., Geike, M., Griep, P., Mahr, P., Rafanelli, M. & Klar, H. 2005 Emulation engine for spiking neurons and adaptive synaptic weights. In *Proc. IJCNN*, pp. 3261–3266.

Hodgkin, A. & Huxley, A. F. 1952 A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **117**, 500–544.

Izhikevich, E. M. 2004 Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* **15**, 1063–1070. (doi:10.1109/TNN.2004.832719)

Izhikevich, E. M. 2005 Simulation of large-scale brain models. www.nsi.edu/users/izhikevich/interest/index.htm.

Izhikevich, E. M. 2006 Polychronization: computation with spikes. *Neural Comput.* **18**, 245–282. (doi:10.1162/089976606775093882)

Jahnke, A., Roth, U. & Klar, H. 1996 A SIMD/dataflow architecture for a neurocomputer for spike-processing neural networks (NESPINN). *MicroNeuro* **96**, 232–237.

Kanerva, P. 1988 *Sparse distributed memory*, Cambridge, MA: MIT Press.

Kohonen, T. 1972 Correlation matrix memories. *IEEE Trans. Comput.* **C–21**, 353–359.

Laughlin, S. B. & Sejnowski, T. J. 2003 Communication in neuronal networks. *Science* **301**, 1870–1874. (doi:10.1126/science.1089662)

Lichtsteiner, P., Posch, C. & Delbruck, T. 2006 A 128×128 120dB 30mW asynchronous vision sensor that responds to relative intensity change. In *Proc. ISSCC*, pp. 508–509.

Lømo, T. 2003 The discovery of long-term potentiation. *Phil. Trans. R. Soc. B* **358**, 617–620. (doi:10.1098/rstb.2002.1226)

Mahowald, M. 1992 VLSI analogs of neuronal visual processing: a synthesis of form and function. Ph.D. dissertation, California Inst. Tech., Pasadena, CA.

Markram, H. 2006 The blue brain project. *Nat. Rev. Neurosci.* **7**, 153–160. (doi:10.1038/nrn1848)

McCulloch, W. S. & Pitts, W. 1943 A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133. (doi:10.1007/BF02478259)

Mead, C. A. 1989 *Analog VLSI and neural systems*. Reading, MA: Addison-Wesley.

Mead, C. A. 1990 Neuromorphic electronic systems. *Proc. IEEE* **78(10)**, 1629–1636. (doi:10.1109/5.58356)

Mehrtash, N., Jung, D., Hellmich, H. H., Schoenauer, T., Lu, V. T. & Klar, H. 2003 Synaptic plasticity in spiking neural networks (SP^{2}INN): a system approach. *IEEE Trans. Neural Netw.* **14**(5), 980–992. (doi:10.1109/TNN.2003.816060)

Moore, G. E. 1965 Cramming more components onto integrated circuits. *Electronics* **38**(8), 114–117.

Mountcastle, V. 1978 An organizing principle for cerebral function: the unit module and the distributed system. In *The mindful brain* (eds G. M. Edelman & V. B. Mountcastle), pp. 7–50. Cambridge, MA: MIT Press.

NEURON 2005 www.neuron.yale.edu/neuron.

O'Keefe, J. & Recce, M. L. 1993 Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus* **3**(3), 317–330. (doi:10.1002/hipo.450030307)

O'Reilly, R. C. 1996 Biologically plausible error-driven learning using local activation differences: the generalized recirculation algorithm. *Neural Comput.* **8**(5), 895–938.

Prange, S. J. & Klar, H. 1993 Cascadable digital emulator IC for 16 biological neurons. In *Proc. 40th ISSCC*, pp. 234–235, 294.

Rall, W. 1959 Branching dendritic trees and motoneuron membrane resistivity. *Exp. Neurol.* **1**, 491–527. (doi:10.1016/0014-4886(59)90046-9)

Saudargiene, A., Porr, B. & Wörgötter, F. 2004 How the shape of pre- and post-synaptic signals can influence STDP: a biophysical model. *Neural Comput.* **16**, 595–625. (doi:10.1162/089976604772744929)

Schoenauer, T., Mehrtash, N., Jahnke, A. & Klar, H. 1998 MASPINN: novel concepts for a neuro-accelerator for spiking neural networks. In *Proc. VIDYNN'98*.

Schwartz, J. & Begley, S. 2003 *The mind and the brain: neuroplasticity and the power of mental force*. New York, NY: Regan Books.

Sivilotti, M. 1991 Wiring considerations in analog VLSI systems, with application to field-programmable networks. Ph.D. dissertation, California Inst. Tech., Pasadena, CA.

Sloman, A. 2004 GC5: The architecture of brain and mind. In *UKCRC grand challenges in computing—research* (eds C. A. R. Hoare & R. Milner), pp. 21–24. Edinburgh, UK: British Computer Society.

Van Rullen, R. & Thorpe, S. 2001 Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Comput.* **13**(6), 1255–1283. (doi:10.1162/08997660152002852)

Werbos, P. 1994 *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. New York, NY: Wiley.

Willshaw, D. J., Buneman, O. P. & Longuet-Higgins, H. C. 1969 Non-holographic associative memory. *Nature* **222**, 960–962. (doi:10.1038/222960a0)

Yang, Z., Murray, A. F., Wörgötter, F., Cameron, K. L. & Boonsobhak, V. 2006 A neuromorphic depth-from-motion vision model with STDP adaptation. *IEEE Trans. Neural Netw.* **17**(2), 482–495. (doi:10.1109/TNN.2006.871711)

Zhu, J. & Sutton, P. 2003 FPGA Implementations of neural networks—a survey of a decade of progress. In *Proc. 13th Ann. Conf. on Field Programmable Logic and Applications*, pp. 1062–1066.