REAL-TIME MILLION-SYNAPSE SIMULATION OF CORTICAL TISSUE

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2013

By
Thomas Sharp
School of Computer Science

Contents

Word Count: 32974

List of Tables

List of Figures

Abstract

The nervous system may be simulated as a network of model neurons as a means to understand the function of the brain. Complex as the mammalian nervous system is, such simulations of any significant scale are computationally and energetically expensive. SpiNNaker is a computer architecture designed to advance the feasible scale of neural tissue models using fifty thousand chips, each containing eighteen low-power processors, to model one billion neurons and one trillion synapses in real-time. This dissertation demonstrates the success of prototype hardware with detailed models of the rodent somatosensory cortex. Simulations are built from neuroanatomical data on a host computer using a simple declarative library of functions, and are executed on SpiNNaker atop an event-driven programming interface that neatly abstracts the intricate details of the machine. Comparisons with reference simulators show that SpiNNaker correctly reproduces established results, and power readings report that each chip draws just one watt during execution. A model of the whisker barrel, derived from the literature, exhibits key responses to simulated stimuli, and a model of the wider barrel cortex, comprising 10^5 neurons and $7{\cdot}10^7$ synapses, demonstrates real-time, massively parallel simulation across 360 processors on 23 chips. Ultimately, SpiNNaker is shown to be an effective architecture for the correct and efficient simulation of neural tissue.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

The four years of my doctoral degree have been amongst the most enjoyable of my life, quite in spite of the tedious obligation to research a thesis. This is entirely due to the people with whom I've been fortunate to work and play.

Steve Furber offered invaluable supervision and independence in equal parts, in order for me to properly experience the successive stages of cluelessness, confidence and cynicism that seem essential to a PhD. I'm grateful for the guidance of Luis A. Plana, upon whose erudite wisdom any sound reasoning of my own is probably based, and Rasmus Petersen, who lent a great deal of critical thought and neuroscientific knowledge. Steve Temple patiently answered innumerable questions on SpiNNaker and stoically repaired all the hardware I clumsily broke. I am finishing this dissertation in reasonable health and manageable debt simply because of Will Toms' repeated council that my work will always be awful and I should just hurry up and submit it. Francesco Galluppi cosuffered countless paper deadlines, conferences and workshops with me; I'm pleased to say that we survived the frequent urges to kill one another and travelled the world together as good friends. The members of the Advanced Processor Technologies group and the School of Computer Science have offered nothing but good humour and strong support, for which I'm sincerely grateful.

Jamie Carruthers is a great friend, whose invitations to leave the lab I should have accepted more frequently. Mum and Dad have given unconditional, invariant love and support without which I'd be anchorless; I can't thank them enough. Likewise Charlotte Constance Wilson, who illuminates everything.

The whole thinking process is still rather mysterious to us, but I believe that the attempt to make a thinking machine will help us greatly in finding out how we think ourselves.

— Alan Turing (1951)

Chapter 1

Introduction

The brain is a computer unlike any other. In animals, exquisite networks of nerve cells synthesise every experience and drive every action. In humans uniquely, brains have long attempted to understand their own function, from the first inquiries into cognition and the self to contemporary descriptions of nervous anatomy and physiology. The electrochemical operation of neurons and synapses is now well characterised, and the structures in which they reside are being mapped in great detail. Yet we still understand very little about the emergent function of these circuits; the cerebral cortex is the substrate of high-level sensory processing and motor control in the brain, and there exists no sound theory of the fundamental computations that take place within. This dissertation presents advances in technology for simulating neural tissue as a means to understand the function of the brain, in the form of large-scale models of the rodent barrel cortex executed on the SpiNNaker computer architecture.

Understanding the function of neural circuits *in vivo* is immensely difficult. The circuits are extremely noisy, and their components are micrometres in size and wired stochastically in three dimensions. In the rodent somatosensory cortex, eighteen thousand neurons in a $200\mu m^3$-volume of tissue process signals from a single whisker. It is impossible to record simultaneously the activity of even one percent of these cells using current electrophysiological techniques, so the view of cortical information processing remains very hazy. Chapter 2 discusses the best understanding of neural function in light of such limited observability, from the basic mechanisms of neural signalling to the current literature regarding cortical anatomy and physiology.

Simulation of neural tissue is a promising alternative methodology for investigating the function of the brain. Neurons may be modelled in terms of their electrophysiological dynamics and simulated in networks with topologies drawn from anatomical data.

If a network model can be shown to reproduce observable *in vivo* phenomena, it may then be used to describe the computations that take place in the simulated tissue. Simulations have significant advantages over biological experiments, in that the former afford greater observability, experimental control and reproducibility. Furthermore, if the mechanisms of neural computation are to be understood then the immense complexity of the nervous system must be represented by abstract models that are amenable to quantitative analysis.

The feasible scale and speed of simulations is currently limited by the enormous parallelism and power disparities between biological and silicon computers. The mammalian brain comprises many billions of neurons and trillions of synapses operating in parallel, whereas supercomputers contain only tens or hundreds of thousands of processors. One processor may simulate very many neurons, but the power requirements of these two computation substrates are still hugely disproportionate: the entire mammalian brain runs on just a few watts, whereas the projected cost of brain-scale simulations on current silicon technology is on the order of gigawatts. Chapter 3 surveys simulation technologies and their limitations, discusses methodologies for efficient simulation, reviews notable simulation studies, and introduces SpiNNaker.

SpiNNaker is a computer architecture designed to simulate up to one billion neurons and one trillion synapses in real-time using little electrical power. The architecture mimics the structure of neural computation with one million low-power processors and a novel interprocessor communication mechanism. Each processor simulates up to one thousand neurons in discrete time and sends and receives simulated action potentials through a local communications controller. The communications controller, in turn, interacts with a packet-switched network that may be programmed to emulate intricate nervous wiring between processors. This dissertation argues that SpiNNaker is an effective simulation technology in terms of flexibility, performance and power, and demonstrates that a significant part of the rat barrel cortex can be modelled on prototypes of the platform.

Massively parallel hardware presents particular software challenges to configuration, run-time programming and simulation specification. For a machine to be useful, it is essential that these tasks are completed correctly and efficiently, and that researchers are able to use the technology without intimate knowledge of the hardware. Chapter 4 discusses the configuration protocols (Sharp *et al.*, 2011a) that are used to boot-up and maintain a SpiNNaker machine, presents an application programming interface (Sharp *et al.*, 2011b) that provides a software abstraction layer for programmers, and

introduces the PyNN and PACMAN interfaces (Galluppi *et al.*, 2012) through which anatomical and physiological data are compiled into SpiNNaker simulations.

SpiNNaker is designed to simulate large models more quickly and efficiently than current technologies. To verify that the hardware meets these design criteria, chapter 5 presents validation, power and performance data. Observations of the boot protocols show that a full-scale machine can be configured in less than one second (Patterson *et al.*, 2012). Comparisons with reference simulators show that SpiNNaker correctly reproduces established results, and performance profiles of the machine- and host-side software explain how the SpiNNaker also outperforms the reference platforms (Sharp and Furber, 2013). Power readings taken during simulation of the cat visual cortex report that each chip consumes a single watt, and so fifty-thousand-chip simulations of a billion neurons are power-feasible (Sharp *et al.*, 2012).

Simulation remains a tentative methodology for research into the brain. To prove the validity of the approach, plausible models of neural tissue must first reproduce activities observed *in vivo*. The rodent barrel cortex is an ideal candidate for such modelling, in that it is exceptionally well characterised anatomically and physiologically. Chapter 6 draws upon the tools and methodology in previous chapters to present simulations of the barrel cortex on SpiNNaker. A series of preliminary experiments attempt to construct biologically plausible networks from a review of data in the literature, with limited success. A coarse model of the whisker barrel reproduces stimulus-responses observed *in vivo* and a model of the broader barrel cortex, comprising 10^5 neurons $7{\cdot}10^7$ synapses, demonstrates real-time, massively parallel simulation across 360 processors on 23 chips. The principal findings of this chapter are that large-scale, detailed models of the cortex are executable on SpiNNaker, but that biological plausibility must be compromised to ensure the dynamical stability of such models (Sharp, Petersen and Furber, in preparation).

The application of computational methods to neuroscience promises enormous advances in the understanding of the brain. Already, automated techniques are reconstructing cells and circuits in three dimensions, and high-performance computers are decoding and interpreting observed activity in the cortex. This dissertation suggests that simulation of neural tissue is also a viable computational methodology for investigating brain function. The hardware and software of SpiNNaker accepts model descriptions from the biological literature and executes large-scale simulations quickly and efficiently. On these foundations a greater understanding of cortical computation may be built: models of neural tissue allow for much freer experimentation that the

living tissue itself, so subsequent experiments may begin to characterise the essence of cortical function in terms of boolean logic, signal processing, Bayesian inference or whichever computational paradigm appears apt. Chapter 7 concludes this dissertation with a review of the presented work, an evaluation of the methodologies employed and the modelling assumptions made, and a discussion of the potential for simulation studies to contribute to neuroscience.

.

The principal contributions of this dissertation are large-scale, biologically inspired simulations of the cerebral cortex that demonstrate the capabilities of SpiNNaker, and the design, implementation and testing of the software upon which these demonstrations are based. Much of this work was conducted in collaboration with other researchers, and parts have been published or submitted for review.

The boot software documented in chapter 4 was developed with Cameron Patterson; the experiments regarding these protocols in chapter 5 were designed, executed and reported upon by the author for presentation at the 2011 International Joint Conference on Neural Networks (Sharp, Patterson and Furber). Partial results of these experiments also appear as a small contribution to a publication in *Journal of Parallel and Distributed Computing* (Patterson, Garside, Painkras, Temple, Plana, Navaridas, Sharp and Furber, 2012).

The application run-time kernel and the application programming interface were designed and implemented in collaboration with the SpiNNaker group and Luis A. Plana respectively. Chapter 4 reproduces work on this topic that was prepared by the author and presented at the 2011 International Conference on Neural Information Processing (Sharp, Plana, Galluppi and Furber).

The initial PyNN implementation for SpiNNaker was conceived and developed by Francesco Galluppi; a subsequent iteration of this software, briefly discussed in chapter 4, was prepared by the author to accommodate the large scale of simulations considered here. Further details appear in a submission to the 2012 International Conference on Computing Frontiers (Galluppi, Davies, Rast, Sharp, Plana and Furber), to which the author made only modest contributions.

The correctness, performance and power requirements of the aforementioned software as executed upon SpiNNaker are examined in chapter 5. A paper accepted to the 2013 International Joint Conference on Neural Networks discusses the former concerns (Sharp and Furber) and another published in *Journal of Neuroscience Methods* exam-

ines the latter (Sharp, Galluppi, Rast and Furber). In each case, the experiments were designed, conducted and reported upon by the author.

The models of the barrel cortex presented in chapter 6 were conceived, researched and simulated by the author. Rasmus Petersen provided considerable guidance through the neuroscientific literature, proposed modelling methodologies and experimental protocols, and assisted in the analysis of results. A paper on the findings of these experiments is in preparation.

In all cases, the contributions of the SpiNNaker group and the supervision of Steve Furber are acknowledged.

.

Francesco Galluppi, Sergio Davies, Alexander D. Rast, Thomas Sharp, Luis A. Plana, and Steve B. Furber. A Hierarchical Configuration System for a Massively Parallel Neural Hardware Platform. In *Computing Frontiers, International Conference on*, pages 183–192, 2012

Cameron Patterson, Jim Garside, Eustace Painkras, Steve Temple, Luis A. Plana, Javier Navaridas, Thomas Sharp, and Steve Furber. Scalable communications for a million-core neural processing architecture. *Journal of Parallel and Distributed Computing*, 72:1507–1520, 2012

Thomas Sharp, Cameron Patterson, and Steve Furber. Distributed Configuration of Massively-Parallel Simulation on SpiNNaker Neuromorphic Hardware. In *Neural Networks, International Joint Conference on*, 2011a

Thomas Sharp, Luis A. Plana, Francesco Galluppi, and Steve Furber. Event-Driven Simulation of Arbitrary Spiking Neural Networks on SpiNNaker. In *Neural Information Processing, International Conference on*, pages 424–430, 2011b

Thomas Sharp, Francesco Galluppi, Alexander Rast, and Steve Furber. Power-efficient simulation of detailed cortical microcircuits on SpiNNaker. *Journal of Neuroscience Methods*, 210:110–118, 2012

Thomas Sharp and Steve Furber. Correctness and Performance of the SpiNNaker Architecture. In *Neural Networks, International Joint Conference on*, 2013

Chapter 2

The Nervous System

The nervous system is an intricate network of cells that allows an animal to sense and interact with its environment. The central nervous system aggregates, processes and stores information from the sensory organs of the peripheral nervous system, and returns signals to the periphery to control the muscles and viscera. The mammalian central nervous system is broadly divided into the brainstem, cerebellum, diencephalon and the cerebral hemispheres. The pathways of the rodent vibrissal system, with which this thesis is primarily concerned, arise in the whiskers and traverse the peripheral nervous system, brainstem and diencephalon to terminate in the cerebral cortex. The latter is a computer that processes sensory stimuli and controls voluntary motor action, and is of significant interest in research into the high-level function of the brain.

The current understanding of the cortical function of somatosensation is presented at the end of this chapter, and is preceded by a review of the neuroanatomical literature on cortical structure. The chapter begins with an introduction to the fundamental aspects of neural signalling: the electrochemical operation of neurons and synapses.

2.1 Neural signalling

The neuron, like all animal cells, is essentially a lipid membrane that encloses a nucleus of genetic data, mitochondria, and various other organelles for controlling cell function. The neuron is distinguished, however, by a membrane that is adapted to the electrochemical signalling mechanism of the action potential. The typical neuron, depicted in figure 2.1, is formed of a tree of dendrites that receives synaptic currents from afferent neurons, a cell soma that integrates such inputs and generates action potentials, and an axon that conveys those signals to efferent neurons (Nolte, 2007).

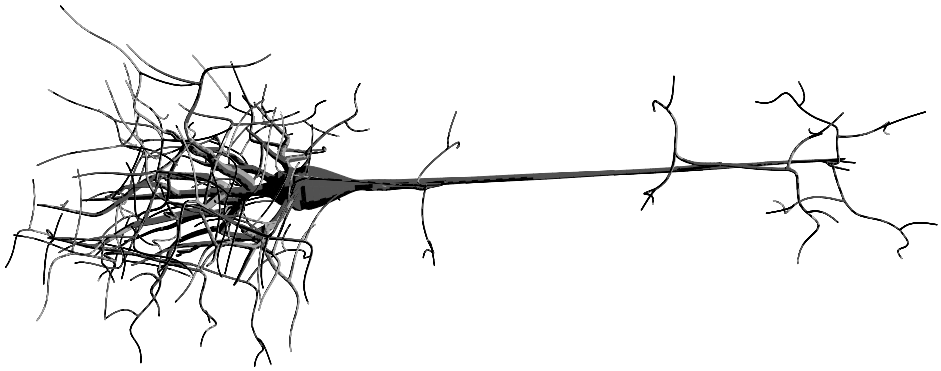


Figure 2.1: Depiction of a typical neuron (Piękniewski, 2013).

2.1.1 Membrane potential

The animal cell membrane is largely impermeable to ions and organic compounds, and thereby maintains an electrochemical gradient between the cytoplasmic and extracellular fluids (Kandel *et al.*, 2000). A chemical concentration of Potassium within the cell, and Sodium without, is established by the *active transport* of ions across the membrane by specialised proteins. Ions also traverse the membrane via protein channels, according to their respective chemical concentration gradients and the overall electrical potential gradient. Considering a single ion species, these currents are balanced at the transmembrane *equilibrium potential*, which is found by the Nernst equation

$$E_{\text{ion}} = \frac{RT}{zF} \ln \frac{[\text{ion}]_\text{o}}{[\text{ion}]_\text{i}}$$

where R is the universal gas constant (8,315mJ/(K°Mol)), T is the temperature in degrees Kelvin, F is Faraday's constant (96,480C/Mol), z is the ion valence, and $[\text{ion}]_\text{i}$ and $[\text{ion}]_\text{o}$ are the concentrations of the ion within and without the cell, respectively (Izhikevich, 2007). Table 2.1 lists the chemical concentrations of Sodium and Potassium in the giant axon of the squid, and the resulting equilibrium potentials at 25°C.

Considering multiple ion species, the overall *resting membrane potential* is a function of both the concentrations of and membrane permeability P_{ion} to each species, which is represented in the Goldman equation

$$V_{\text{rest}} = \frac{RT}{F} \ln \frac{P_\text{K}[\text{K}^+]_\text{o} + P_\text{Na}[\text{Na}^+]_\text{o}}{P_\text{K}[\text{K}^+]_\text{i} + P_\text{Na}[\text{Na}^+]_\text{i}}.$$

Evaluating the equation with the parameters in table 2.1 gives a resting membrane potential of -60mV: a *hyperpolarised* state that is significantly closer to the equilibrium

Ion species	$[\mathbf{ion}]_\mathbf{i}$	$[\mathbf{ion}]_\mathbf{o}$	$E_\mathbf{ion}$	$P_\mathbf{ion}$
K^+	400	20	-75	1.00
Na^+	50	440	+55	0.04

Table 2.1: Electrochemical properties of the squid axon (Kandel *et al.*, 2000).

potential of Potassium than of Sodium, due to the greater permeability of the membrane to the former. However, P_{ion} varies for each ion species as a function of both membrane potential and time, such that a slight influence from an external current source may cause a rapid, radical change in conditions.

2.1.2 The action potential

The action potential is a brief, transient *depolarisation* of the cell membrane, caused by synaptic currents or artificial stimulation. In describing these dynamics, it is useful to represent the constituent ionic currents in the equivalent-circuit form

$$I_{\text{ion}} = g_{\text{ion}}(V - E_{\text{ion}})$$

where instantaneous current for a particular ion species is given by the difference between the equilibrium and membrane potentials, multiplied by the membrane conductance g_{ion}, which is proportional to permeability. Membrane potential dynamics may then be represented as a sum of ionic currents

$$C\frac{dV}{dt} = -g_{\text{K}}(V - E_{\text{K}}) - g_{\text{Na}}(V - E_{\text{Na}}) + I$$

where C is the membrane capacitance, approximately 1.0μF/cm^2 with cell membrane area on the order of 0.1mm^2, and I is an external current source (Izhikevich, 2007). Hodgkin and Huxley (1952) observe that the time- and voltage-dependent (in)activation of transmembrane channels varies membrane conductances and thereby gives rise to the action potential, which is ultimately described as

$$C\frac{dV}{dt} = -\bar{g}_{\text{K}}n^4(V - E_{\text{K}}) - \bar{g}_{\text{Na}}m^3h(V - E_{\text{Na}}) - g_{\text{L}}(V - E_{\text{L}}) + I$$

where $\bar{g}_{\text{ion}}$ is the maximum membrane conductance, n^4 is the proportion of K^+-conductant proteins that are *activated* (open), m^3 denotes the same for Na^+, and h is the

proportion of Na^+-conductant proteins that are *deinactivated* (not blocked). $g_L(V-E_L)$ denotes a *leak current* of Chloride, for which conductance is constant. The activation variables are in turn dynamic

$$\frac{dn}{dt} = \alpha_n(1-n) - \beta_n n$$

$$\frac{dm}{dt} = \alpha_m(1-m) - \beta_m m$$

$$\frac{dh}{dt} = \alpha_h(1-h) - \beta_h h$$

and depend on the empirically tuned functions

$$\alpha_n = 0.01\frac{10-V}{\exp\left(\frac{10-V}{10}\right)-1} \qquad \beta_n = 0.125\exp\left(\frac{-V}{80}\right)$$

$$\alpha_m = 0.1\frac{25-V}{\exp\left(\frac{25-V}{10}\right)-1} \qquad \beta_m = 4\exp\left(\frac{-V}{18}\right)$$

$$\alpha_h = 0.07\exp\left(\frac{-V}{20}\right) \qquad \beta_h = \frac{1}{\exp\left(\frac{30-V}{10}\right)+1}$$

with typical equilibrium potentials and maximum conductances of

$$E_K = -77\text{mV} \qquad g_K = 36\text{mS/cm}^2$$

$$E_{Na} = 55\text{mV} \qquad g_{Na} = 120\text{mS/cm}^2$$

$$E_L = -54.4\text{mV} \qquad g_L = 0.3\text{mS/cm}^2$$

Figure 2.2 shows the response of the Hodgkin-Huxley neuron model to a small, brief depolarising current, and provides the basis for a qualitative description of the model dynamics. The input current at $t = 2$ms (top, I) increases membrane potential, which causes fast activation and slow inactivation of Sodium channels (middle, rising m and falling h). In the brief interval between activation and inactivation, conductance to Sodium increases sharply at $t = 4ms$ (bottom, g_{Na}) so that membrane potential moves towards $E_{Na} = 55$mV, in accordance with the Goldman equation. The consequent influx of Sodium to the cell down the electrochemical gradient increases V, which further increases g_{Na}, and from this positive feedback a membrane potential 'spike' (top) rapidly follows. Positive membrane potential activates Potassium channels

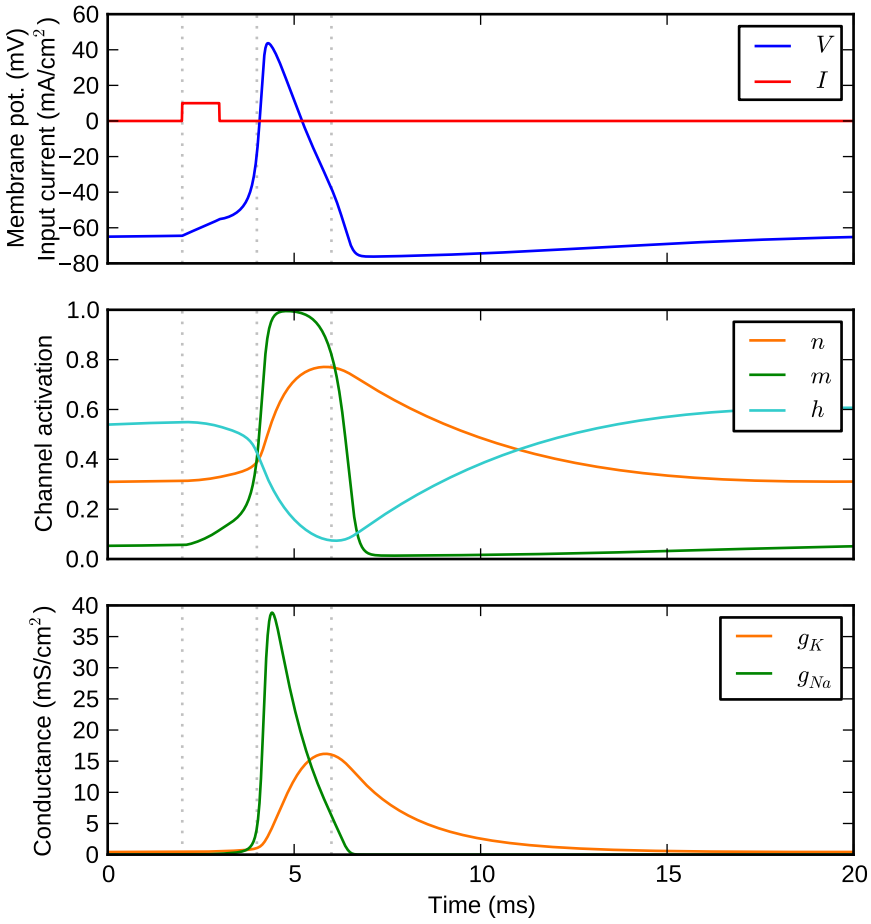


Figure 2.2: Response of the Hodgkin-Huxley model to a brief depolarising current.

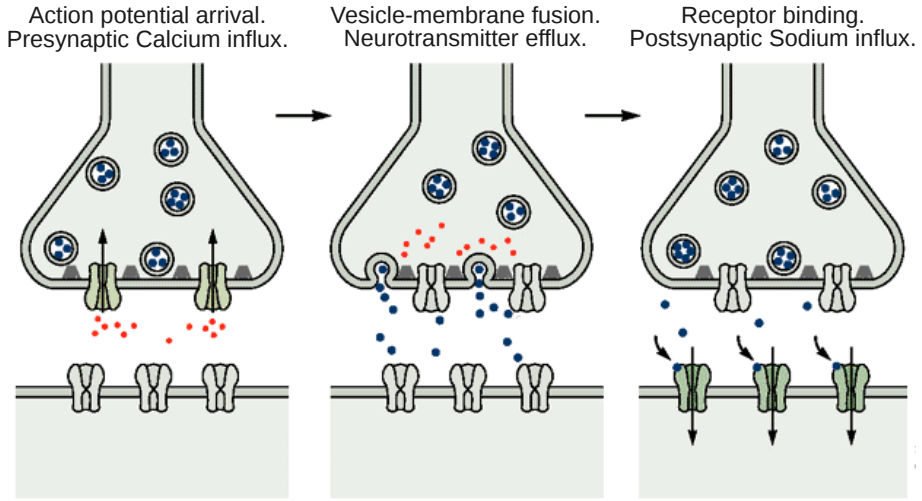


Figure 2.3: Chemical signalling in the synapse. Adapted from Kandel *et al.* (2000).

around $t = 6$ms (middle, rising n) and the efflux of the ion from the cell down the electrochemical gradient, in conjunction with the inactivation of Sodium channels, repolarises the membrane. The cell then undergoes a *refractory period*, during which it may not spike, as the activation variables return to their resting values. Finally, although not represented in this model, Sodium that diffuses across the membrane at a particular site carries a charge that also flows *along* the membrane through the cytoplasm, thereby depolarising neighbouring areas of membrane. Thus, an action potential formed in the soma regeneratively propagates down the axon, without loss of signal power. A mathematical description of this process and further details of the Hodgkin-Huxley model are presented by Dayan and Abbott (2001).

2.1.3 Synaptic signalling

The synapse is a junction between neuron membranes that may form when axons and dendrites pass within some micrometres of one another. As the axon of one neuron terminates upon the synapses of others, so do its action potentials.

Figure 2.3 shows the response of an excitatory synapse to an action potential. Upon arrival, the action potential opens Calcium channels in the presynaptic membrane terminal, and Ca^{2+} flows into the cell. Consequently vesicles containing neurotransmitters are drawn and fused to the presynaptic terminal, and their contents are released into the synaptic cleft. The transmitters bind to receptors on Sodium channels in the postsynaptic terminal, which open to permit and influx of Na^{+} and thereby cause a postsynaptic depolarisation, or *excitatory postsynaptic potential*. An analogous pro-

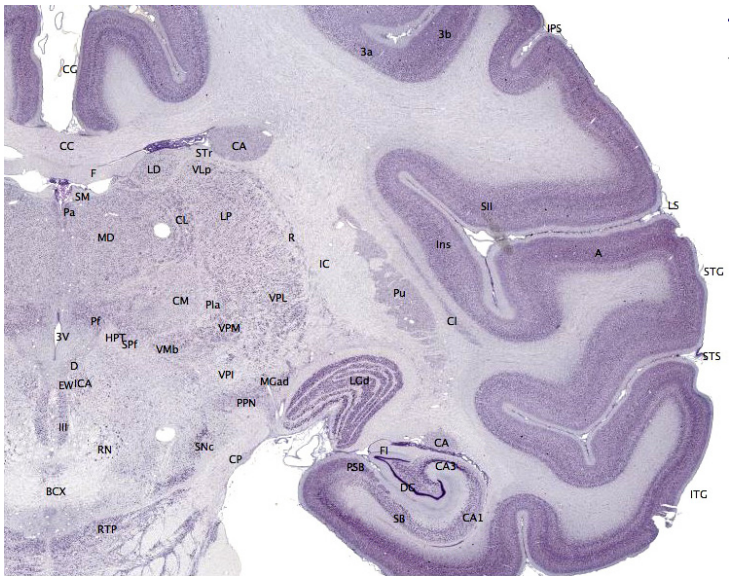


Figure 2.4: Stained coronal section of macaque brain (brainmaps.org, 2013).

cess, using different transmitters and channels, occurs across inhibitory synapses that induce postsynaptic hyperpolarisations. Kandel *et al.* (2000) detail these chemical processes, discussion of which is omitted here for brevity.

The sum of all such postsynaptic currents forms the inputs to a neuron, represented by I in the Hodgkin-Huxley model. Synaptic currents therefore determine the outputs of neurons, which in turn form the inputs to other neurons, and so on *ad mortem*. So, activity in the nervous system depends upon extrinsic input, the dynamics of neurons and synapses and, crucially, the intrinsic system structure.

2.2 Cortical anatomy and physiology

The cerebral cortex is the corrugated husk of the cerebral hemispheres, and is the structural substrate of high-level computation in the brain. Figure 2.4 shows a coronal section of the macaque brain, with the purple-stained sulci and gyri of the auditory cortex on the right and, across the *white matter*, subcortical structures including the ventral posteromedial nucleus (VPM) of the thalamus on the centre-left. Sensory stimuli arrive at the cortex via the axonal projections of thalamic neurons, which are in turn driven by the sensory organs via the trigeminal nucleus. Pathways from the sensory organs to the thalamus are neither visible in figure 2.4 nor significantly featured in the following chapters, but Bosman *et al.* (2011) offer a comprehensive review of those involved in the rodent whisker system.

In the sensory cortices, neurons are arranged into a macroarchitecture of six layers parallel to the cortical plane, numbered in ascending order from external to internal. Layer 4 is termed the *granular layer*, and the remaining layers are described relatively: layers 1, 2, and 3 comprise the *supragranular layers*, and layers 5 and 6 are the *infragranular layers*. The axonal projections of excitatory cells dictate the flow of information in the thalamocortical loop: thalamic axons primarily innervate the granular layer, which in turn makes projections into the supragranular layers; cells in these layers project laterally within the layer and radially to the infragranular layers, which project through the white matter to other (sub)cortical areas; all layers make recurrent excitatory projections, and the level of excitatory activity is controlled by local networks of inhibitory cells (Thomson *et al.*, 2002; Thomson, 2003; Thomson and Lamy, 2007).

2.2.1 Methodology

The microarchitecture of the cortex is immensely complex. A cubic millimetre of tissue contains on the order of 10^5 neurons and 10^8 synapses (Wimmer *et al.*, 2010; Meyer *et al.*, 2010a;b). This complexity impedes a complete cartography of the cortex, and the best understanding of its wiring is currently obtained by inference from somewhat sparse observations.

Cell counts for different areas and layers are made *in vitro* from tissue that is fixed, sliced and stained. Braitenberg and Schüz (1991) provide an early statistical description of cortical tissue through manual counts of cell somata, upon which Oberlaender *et al.* (2011) improve with automated computer vision techniques applied to rodent whisker barrel slices. Such techniques are also capable of reconstructing complete morphologies of individual cells from successive slices, which offers some insight into cell physiology and potential connectivity; Binzegger *et al.* (2004), for example, draw an extremely detailed map of cat visual cortex by inferring synaptic connectivity from the overlap of reconstructed axons and dendrites.

Electrode recordings *in vivo* and *in vitro* supplement static maps with vital data on the physiology of functioning neural tissue. Single electrode recordings are used to investigate membrane properties in current- and voltage-clamp experiments like those used by Hodgkin and Huxley (1952) and paired- or multielectrode recordings are used to describe the synaptic connectivity of cells by inducing a spike in a presynaptic candidate and looking for a postsynaptic potential in others (Lefort *et al.*, 2009). *In vivo* multielectrode techniques have the advantage of reporting definite synaptic connectivity and postsynaptic potential properties in behaving animals, both of which are lacking

in the inferred-connectivity data of Binzegger *et al.* (2004).

Optical stimuli may be substituted for presynaptic electrodes in these experiments. Packer and Yuste (2011) genetically modified mice to express light-sensitive inhibitory neurons in the cortex, and used a scanning light source on tissue slices to identify the presynaptic partners of target cells recorded with electrodes. With such techniques, the sampling of presynaptic candidates is precisely controlled, and successive candidates may be examined without repetitive adjustment of a presynaptic electrode. Optogenetics offers unparallelled spatial and temporal resolution in the control of tissue, and is likely to become a fundamental technique in future neurophysiology.

2.2.2 Data

All anatomical and physiological techniques applied to the cortex involve painstaking experiments to build dense maps of small tissue volumes or sparse maps of larger samples. Considered in conjunction, these studies provide a broader picture of the lateral (planar) and radial (transplanar) microarchitecture of the cortex.

Lateral organisation

There is a great deal of contention over the form of lateral connectivity in the cortex. Studies that report discrete patches of neurons with common stimulus-response properties, or *receptive fields*, suggest the existence of distinct functional *columns*, but no structural analogue of the column has been clearly identified.

Mountcastle (1956) provides evidence for cortical columns, on the basis of data from single-electrode experiments. Penetrations perpendicular to the surface of the somatosensory cortex recorded from cells with common receptive fields at all depths, and tangential penetrations recorded from neurons with different receptive fields at different depths. On the basis of these findings, Mountcastle posits that neurons in the cortex are organised into radial columns, each with particular receptive fields. However, da Costa and Martin (2010) argue in agreement with Horton and Adams (2005) that no structural basis for the column has been definitely observed in the cortex. da Costa and Martin note that the apical dendrites of pyramidal neurons do form radial bundles that may be described as minicolumns, but the basal dendrites show no such grouping and neurons within these putative minicolumns are as likely to be connected to cells in neighbouring minicolumns as their own. Indeed, the proximal synaptic clusters of the identified pyramidal cells span distances over which many different receptive fields

are recorded, and the distal clusters have been shown to innervate dendrite bundles of cells with similar receptive fields (Binzegger *et al.*, 2007). This diffuse-local and specific-distal connectivity clearly contrasts with the notion of hard boundaries in the lateral organisation of the cortex.

Alonso and Swadlow (2005) address the question of how, then, receptive fields are synthesised, with respect to layer 4 neurons of the visual and somatosensory cortices. Certain mammals have extremely high visual resolution, which suggests that there must be a strong correlation between the spiking of retinal, thalamic and cortical cells. Thalamic neurons receive input from just a few retinal cells, each layer 4 excitatory neuron receives only around thirty thalamic afferents, and connected thalamic and excitatory cells do exhibit strongly correlated receptive fields. In contrast, inhibitory neurons receive inputs from thalamic cells of numerous different sensory field receptivities, and respond to a broader range of stimuli as a result. So, Alonso and Swadlow argue that distinct receptive fields arise dynamically from the response of *specific excitatory* cells to their respective thalamic innervators, and that *broadband inhibitory* cells provide contrast in the cortical receptive field by suppressing activity in populations of off-stimuli excitatory cells.

Lateral structure, however, *is* unambiguously apparent in the barrel cortex of whisking animals (Woolsey and der Loos, 1970; Petersen, 2007). Here, discrete *barrels* in the granular layer correspond one-to-one with whiskers, and are clearly visible under appropriate staining; figure 2.5 shows a coronal slice of rat brain in which fluorescent dye in thalamic axons clearly identifies the thalamus at the bottom right, and barrels at the top left (Wimmer *et al.*, 2010). Furthermore, composition and connectivity within the barrel appears to be laterally homogeneous, and the anatomical and physiological parameters of the tissue are well characterised (Lefort *et al.*, 2009). The rodent barrel cortex is therefore an ideal candidate for simulation.

Radial organisation

The *barrel column* is the notional projection of the whisker barrel into the supra- and infragranular layers. Wimmer *et al.* (2010) present the dimensions of the rat barrel column in terms of the thalamocortical projection domain, which was mapped *in vitro* by virus-mediated expression of different fluorescent proteins in axons from the ventro posteromedial (VPM) and posteriomedial (POm) nuclei. The data show that columnar innervation domains are formed by the VPM axons that target barrels, and by the POm axons that target interbarrel septa, although separation is less clear in the infragranular

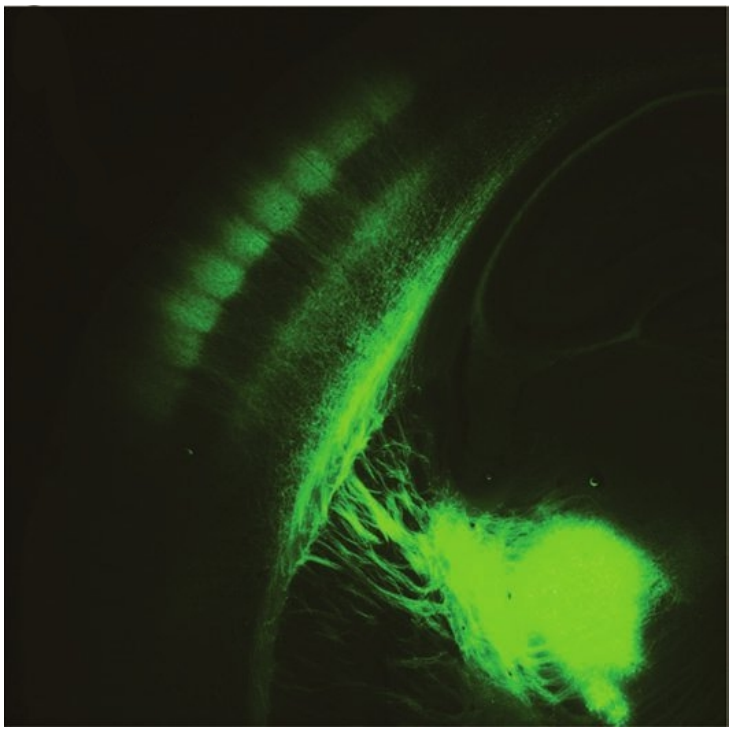


Figure 2.5: Barrels in rodent cortex. Adapted from Wimmer *et al.* (2010).

layers. Defined by these thalamocortical projections, the average column has a tangential area of 0.12 millimetres squared and a depth of 1.84 millimetres. In a sequential article, Meyer *et al.* (2010a) report the number of neurons in the average barrel column: an exhaustive automatic count of all somata in slices of a 1.5 millimetre-cubed volume of cortex identified 17,560 $\pm$ 399 neurons per column, of which 0.15$\times$ are thought to be inhibitory. As reported in the final article of the series, Meyer *et al.* (2010b) used these data to estimate the distribution of thalamic synapses upon the barrel column: the authors automatically reconstructed the morphologies of eighty two cortical cells from sliced tissue, and inferred synapse positions from neuron counts and the presumed overlap of thalamic axons and cortical dendrites. In contrast to the classic thalamocortical loop described by Thomson and Lamy (2007) Meyer *et al.* suggest that excitatory neurons in all layers receive between one hundred and one thousand synapses from the thalamus; nevertheless, the strongest pathway between thalamus and cortex does appear to be, as expected, between VPM and layer 4 neurons. Oberlaender *et al.* (2011) employ the same methods and data to further map the circuit between VPM and cortical excitatory cells: somata in the thalamic nuclei were counted, and electrodes were used to record the spiking properties of barrel cells *in vivo* before the tissue was sliced and the axons and dendrites neurons reconstructed. The authors show that the dendritic

trees of barrel cells precisely overlap the axonal arbours of the 285 ± 13 neurons identified in each VPM barreloid, and that dendrite morphology and VPM-synapse-count correlates with, respectively, spontaneous and whisker-evoked spiking rates of barrel cells. According to Bruno and Sakmann (2006) each excitatory cell in the granular layer receives synapses from around half of the neurons in the corresponding barreloid.

Lefort *et al.* (2009) describe the physiology of the mouse barrel column in some detail, using multiple simultaneous electrode recordings in the C2 barrel to map definite, rather than inferred, synaptic connectivity. In 322 slices from 307 animals, 2,550 excitatory neurons were tested for synaptic connections with one another. A total of 8,895 pairs were tested by eliciting presynaptic spikes and observing (the absence of) postsynaptic potentials, and 909 synaptic connections were found. The study yields three substantial datasets: the electrophysiological properties of excitatory neurons in each layer, including resting membrane potentials, membrane resistances, membrane time constants and action potential thresholds; the probabilities of synaptic connectivity between cells in each layer, which are in broad agreement with the classical thalamocortical loop described by Thomson and Lamy (2007); and the time courses and amplitude distributions of postsynaptic potentials, which are remarkable. The latter data show that the postsynaptic potential amplitudes follow an exponential distribution: a small number of synapses drive large postsynaptic potentials, but most synapses elicit little postsynaptic response. Furthermore, the coefficients of variation of postsynaptic potentials across multiple trials also follow an exponential distribution: strong synapses elicit potentials of a regular and reliable amplitude, but weak synapses cause highly variable postsynaptic responses. The authors suggest that the many weak unreliable synapses are an appropriate substrate for functional rewiring under activity-dependent synaptic plasticity, and that the few strong reliable synapses are important for the propagation of information in sparsely active networks. Avermann *et al.* (2012) continued this work, with the aid of optogenetics, to map the supragranular inhibitory network of the barrel column. They show that synapses between excitatory and inhibitory neurons are both more probable and stronger than between excitatory neurons alone.

Sun *et al.* (2006) note that both excitatory and inhibitory neurons are activated by thalamic stimuli, and hypothesise that strong inhibition is necessary to prevent hyperactive excitation and to discretise responses to sensory inputs. To examine the inhibitory circuits of the rat barrel column, the authors tested seventy pairs of cells *in vitro* and found nine synaptic connections from excitatory to inhibitory neurons and fifteen in the opposite direction. In this small dataset, inhibitory postsynaptic potentials

on excitatory cells have an amplitude of 6.3 ± 1.1 millivolts, which is almost six times greater than the excitatory effect on inhibitory cells. Furthermore, the decay time of the average inhibitory potential is four times greater than its excitatory counterpart. These data point to a strong system of inhibition that controls the excitatory response to sensory stimuli, which is confirmed by observations that excitatory neurons receive significant inhibition from just a few cells soon after thalamic stimulation.

Kätzel *et al.* (2011) present a more comprehensive survey of inhibitory networks in all layers of the mouse somatosensory, visual and motor cortices. Optogenetic techniques were used to elicit action potentials in inhibitory cells, and electrophysiological recordings were made of postsynaptic potentials in excitatory cells. In fifty four successive trials in coronal slices of somatosensory cortex, an excitatory cell was identified and voltage clamped. The slice was scanned by a laser in order to evoke spikes in inhibitory cells, and the timing of inhibitory postsynaptic potentials in the excitatory target was then used to infer the somatic location of each inhibitory source. The authors find that lateral inhibitory projections are constrained to a single barrel in the granular layer, but extend up to five hundred micrometres in other layers, thereby permitting interbarrel inhibition. Radial inhibitory projections are generally weak in the somatosensory cortex, but there are some synapses from the granular to the supra- and infragranular layers. Kätzel *et al.* find significant differences between the structure of certain projections in the examined sensory areas, which contrasts with the notion of a canonical microcircuit that is invariant throughout the cortex (Binzegger *et al.*, 2004; Izhikevich and Edelman, 2008).

Packer and Yuste (2011) further examine the lateral extent of inhibition in the supra- and infragranular layers, using similar optogenetic techniques. In coronal slices of mouse cortex 350 micrometres thick, monosynaptic connections from inhibitory to excitatory neurons were characterised in terms of latency, amplitude and time constant. Results show that local connectivity is extremely dense, such that half of all cell pairs within two hundred micrometres are synaptically connected. The probability of connection decays exponentially, with a decay constant of 124 micrometres in the supragranular layers and 183 in the infragranular. There is no correlation between the interconnectivity of excitatory cells and the probability of inhibitory synapses upon them, but there are spatial patterns of connectivity such that, for example, inhibitory synapses onto excitatory cells in the supragranular layers are more likely to originate from cells closer to the pia. Packer and Yuste report a greater synaptic density than previous publications, but are confident that their methodology gives an accurate pic-

ture of connectivity. Thus, the authors argue that inhibitory cells effectively provide 'blanket inhibition' on all nearby excitatory cells.

Miscellanea

The work of Binzegger *et al.* (2004) is noteworthy in the literature on cortical anatomy. The authors present an extremely detailed map of cat visual cortex using an inferential technique that has since been employed to significant effect by Meyer *et al.* (2010b) amongst others. Binzegger *et al.* made three-dimensional reconstructions of thalamic axons and thirty nine neurons in slices of cat visual cortex, and classified the cells according to soma location and dendrite morphology. The average dendrite length of cells in each class was surmised from the reconstructions, and synapses on a sample of axons were counted under an electron microscope. These data were used to infer a cortical map on the assumption that axonal synapses distribute uniformly across the dendrites in the volume of tissue through which they pass. A variant of Peters' rule (Peters and Feldman, 1976) was used to determine the average number of synapses $\bar{s}_{ij}^u$ formed in layer u upon each neuron of class i by all neurons of class j

$$\bar{s}_{ij}^u = (1 - \rho_j^u) S_j^u \frac{d_i^u}{\sum_k n_k d_k^u} + \rho_j^u S_j^u \frac{\delta_i^u}{N^u}$$

where ρ_j^u is the proportion of synapses formed by neurons of class j directly upon cell bodies; S_j^u is the total number of synapses formed by neurons of class j in layer u, which is the product of the number of neurons n_j and the average number of synapses $\bar{s}_j^u$ formed in layer u by each neuron; d_i^u is the average length of dendrite formed in layer u by neurons of class i; $\sum_k n_k d_k^u$ is the total length of dendrites in layer u; δ_i^u is 1 if the cell soma of neuron class i resides in layer u and 0 otherwise; and N_u is the total number of neurons in layer u. Binzegger *et al.* took n_j from the literature, and found $\bar{s}_j^u$ and d_i^u by the methodology described above. The resulting data describes the typical thalamocortical loop, but also suggests that a significant proportion of the 5,651 $\pm$ 3,120 synapses formed on a neuron come from many weak, disparate sources.

It is worth observing that Perin *et al.* (2011) find significant structures in the cortex that are not represented in the data of, for example, Lefort *et al.* or Binzegger *et al.*. The latter implicitly assume that, for a given source and target population, the probability of a synapse existing between any cell pair is conditionally independent. However, Perin *et al.* show that the synaptic probability between a cell pair is proportional to the number of common neighbours, so pyramidal neurons in the cortex form small-world

networks with neural clusters comprising tens of cells. They suggest that these clusters may form computational primitives from which higher-order structures are build, and Tomm (2012) confirm that such structure is significant in the computational function of cortical models.

There are innumerable further anatomical studies that cannot be considered here. However, it should be noted that Thomson (2003) discusses interlaminar synapses in the cortex in some detail, and Thomson and Lamy (2007) offer a comprehensive review and tabulation of corticoanatomical data from different areas and animals; Gupta *et al.* (2000) elaborate on the arrangement of inhibitory neurons in the cortex; and Stepanyants *et al.* (2008) propose another inferred-connectivity metric that derives from the proximity of axons and dendrites, which they argue varies less over time than the presence of individual synapses.

2.3 Cortical function

Distinct areas of the cortex perform functions of vision, audition, language and motor control, amongst others. Beyond the basic responsibilities of each area, little is known about how the cortex uses action potentials for computation, but the cortical response to sensory stimuli is best understood in the rodent whisker system.

Rodents brush their whiskers rhythmically across surfaces in their environment to build a spatial map. Signals from each whisker are conveyed via discrete barreloids in the thalamus to discrete barrels in the granular layer of the somatosensory cortex. Unitary whisker stimuli only have parameters of deflection amplitude, speed and direction, so in the barrel cortex it is possible to precisely correlate cortical activity with particular sensory input.

Simons and Carvell (1989) investigate the response of the rat cortical barrel to deflection of its one principal and four adjacent whiskers. In successive trials with anaesthetised rats, the authors recorded 135 cells in a thalamic barreloid, and 242 excitatory and 16 inhibitory cells in the corresponding barrel. The principal and adjacent whiskers were deflected and the responses were observed. The authors describe four transformations between the responses of thalamic and excitatory barrel neurons: thalamic cells have greater levels of spontaneous spiking than excitatory cells; excitatory cells respond with different numbers of spikes to the onset and offset of whisker deflection, whereas thalamic cells do not; excitatory cells respond weakly to deflection of an adjacent whisker, unlike thalamic cells; and the response of excitatory cells to principal

whisker deflection is suppressed if it is immediately preceded by deflection of an adjacent whisker, despite homogeneous thalamic responses. Inhibitory barrel cells, in contrast, do not exhibit these response transformations, and behave similarly to thalamic cells. This has the effect that stimulation of an adjacent whisker raises inhibitory firing in the barrel, such that the excitatory response to subsequent stimulation of the principal whisker is suppressed. The activity of the barrel cortex may therefore represent the time difference of deflections to neighbouring whiskers, from which spatial properties of the environment can be inferred using knowledge of whisker position. Simons and Carvell hypothesise that this function arises from four organisational principles: nonlinear stimulus-responses of excitatory and inhibitory neurons; greater responsiveness of inhibitory neurons to input; intrabarrel connections between cortical cells; and concurrent stimulation of both excitatory and inhibitory cells by the thalamus.

Bruno and Sakmann (2006) describe exactly how the thalamus conveys signals to the cortex. The question is particularly interesting because projections from the thalamus comprise only $0.15\times$ of all synapses upon excitatory neurons in the granular layer. It has been hypothesised that thalamocortical synapses must be particularly strong to drive cortical activity, or that recurrent connections in the cortex amplify thalamic signals (Binzegger *et al.*, 2009). Bruno and Sakmann argue, to the contrary, that individual thalamic synapses are weak, and that it is their convergent and synchronous innervation of granular cells that makes information transmission possible. The authors recorded simultaneously from cells in the rat thalamus and cortex, and stimulated whiskers with small periodic deflections. The properties of unitary and total postsynaptic potentials were inferred from multiple recordings, and the structural connectivity between thalamus and cortex was established. Their data show that single whisker stimulus causes an immediate fifteen millivolt fluctuation in cortical neuron membrane potential, and that one thalamic synapse contributes just half a millivolt to this figure. The cortical response is too fast to be a consequence of recurrent amplification, so it must driven by the synchronous firing of around thirty thalamic cells. There are approximately two hundred cells in one thalamic barreloid, and Bruno and Sakmann find seventeen of forty thalamocortical pairs connected, so eighty five synapses are expected to contact each cortical cell. The authors consider this number to be approximately in line with their predictions, and consequently state that thalamus alone can drive a cortical sensory response.

de Kock *et al.* (2007) further investigate the cortical response to thalamic stimulation, aiming to characterise the layer and cell-type responses to single whisker deflec-

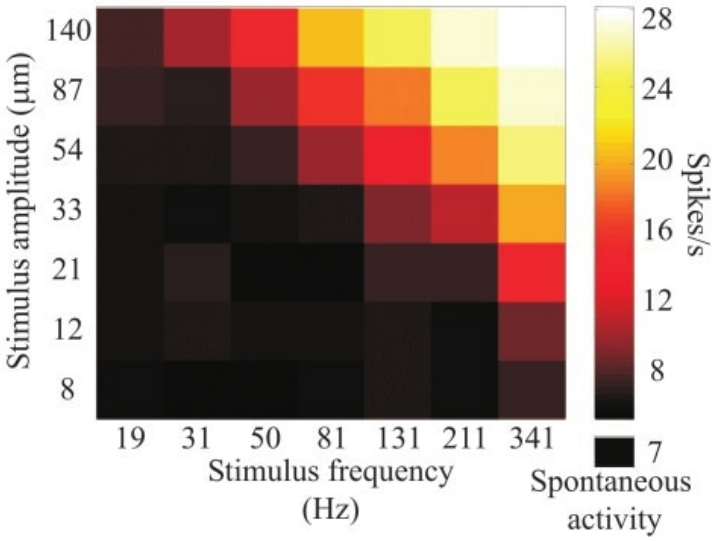


Figure 2.6: Barrel cortex stimulus-response (Arabzadeh *et al.*, 2003).

tions in the rat. The authors recorded juxtasomatically from seventy seven excitatory cells in all cortical layers, and performed *post hoc* reconstructions to identify cell morphologies and laminar positions. The stimulus-response in all layers was characterised as the difference between evoked and spontaneous activity in a post-stimulus window of one hundred milliseconds. Contrary to the view that the thalamus primarily innervates the granular layer, de Kock *et al.* report immediate responses in all layers: supragranular cells responded with .11 $\pm$ .14 action potentials per stimulus, granular cells with .41 $\pm$ .41, infragranular layer 5 slender-tufted cells with .15 $\pm$ .35, infragranular layer 5 thick-tufted cells with .64 $\pm$ .47, and infragranular layer 6 cells with .31 $\pm$ .35 action potentials per stimulus. The strong responses of particular infragranular cells suggest that the thalamus indeed targets these layers. The authors also show that the response magnitude in the higher layers is proportional to the spontaneous spiking in these layers: the granular layer, which has high spontaneous activity, responds more strongly than the supragranular layer, which is largely quiescent in the absence of input. The inverse relationship is found between response latency and spontaneous spiking, suggesting that spontaneous activity primes membrane potentials close to threshold for a fast, strong response to stimuli. Ultimately, the authors infer that each stimulus is represented sparsely by less than one action potential per neuron in a layer- and cell-type specific manner.

Arabzadeh *et al.* (2003; 2004) describe what is represented by the cortical response to continuous whisker stimuli. The authors delivered sinusoidal vibrations with seven am-

plitude (A) and seven frequency (f) variations to individual whiskers of anaesthetised rats, and recorded activity in the granular layer of the corresponding barrel. The number of spikes per second in a post-stimulus window of five hundred milliseconds were counted and plotted against the stimulus parameters, as in figure 2.6. The results show that barrel spiking correlates with neither dimension independently, but rather corresponds to the product of the stimulus parameters Af. This implies that the rat barrel represents only the speed of whisker motion during exposure to textured surfaces, and is therefore unable to discriminate between two stimuli $A_1 f_1 = A_2 f_2$ that are equal in this regard. Adibi *et al.* (2012) corroborate these results with experiments on behaving rats. An environment was created with a nose-poke hole bordered on the left and right by vibrating pads and corresponding water tubes; when the rats poked their noses into the hole, the left and right pads vibrated at some speed $(A_l f_l, A_r f_r)$ and a reward was offered at the water tube on the side of greater amplitude. Adibi *et al.* show that rats become quite competent at detecting the higher amplitude and receiving the reward when f_l is equal to f_r. However, when the increased amplitude of, say, the left pad is offset by an increased frequency in the right, such that $A_l f_l = A_r f_r$, the rats' discriminatory performance falls to chance. The authors therefore confirm that the barrel cortex represents surface textures in terms of the compound parameter Af, which is proportional to whisker vibration speed.

2.4 Summary

Neural computation is rather remarkable in both form and function. Neural signalling is an intricate electrochemical process that exploits the selective ionic permeability of cell membranes to transmit pulses of information between cells. In the cortex these cells are wired into exquisite synaptic circuits that are structured at varying degrees of scale. The cortical plane is divided laterally into areas responsible for vision, audition, language, somatosensation and motor control, but all sensory areas clearly exhibit a common laminar structure: signals arrive into the granular layer from the sensory organs via the thalamus, and follow a typical loop through the supra- and infragranular layers, before laterally traversing or leaving the cortex. In the rodent cortex, clear barrels in the granular layer correspond one-to-one with individual whiskers, and so whisking animals are ideal subjects for investigating the relationship between tissue structure and function. Studies of anaesthetised and behaving rodents suggest that the barrel cortex represents environmental space through differential responses to suc-

cessive deflections of principal and adjacent whiskers, and represents the texture of encountered surfaces by a firing rate corresponding to the product of whisker vibration amplitude and frequency.

Chapter 3

Simulating the Nervous System

Simulation of neural tissue is a promising methodology for research into the function of the brain. Indeed, if the mechanisms of neural information processing are to be understood at all, the immense complexity of the nervous system must be represented by abstract models that are amenable to quantitative analysis. One such approach is to model membrane potentials and synaptic currents as differential equations that, when simulated as a network of cells, emulate the collective dynamics of their biological counterparts. This approach affords perfect tissue observability, experimental control, and reproducibility, which cannot be said of experiments *in vivo*.

This chapter introduces mathematical abstractions of neurons and synapses, and methodologies for their simulation. Section 3.2 makes a comprehensive review of existing simulation technologies, including SpiNNaker, and section 3.3 discusses notable simulations of the visual and somatosensory cortices.

3.1 Simulating neurons and synapses

Hodgkin and Huxley (1952) present the first detailed model of action potential genesis and propagation in the neuron. Solved numerically, the model equations precisely describe observed current flows in the giant axon of the squid. However, in capturing the detailed biophysics of the action potential, the model is extremely computationally expensive. Around one thousand operations are required to simulate one neuron for one millisecond (Izhikevich, 2004) so the Hodgkin-Huxley formalism is unsuited to large-scale experiments.

The leaky integrate-and-fire neuron (LIF) model represents the cell membrane rather more simply as a single constant *leak* conductance and an input current

$$c_m \frac{dV}{dt} = -g_L(V - E_L) + \frac{I}{A}$$

where A is the membrane area, $c_m = \frac{C_m}{A}$ is the specific membrane capacitance, g_L is the leak conductance, E_L is the equilibrium potential, and I is the input current (Dayan and Abbott, 2001). The equation may be multiplied through by the specific membrane resistance $r_m = \frac{1}{g_L}$ to give

$$\tau_m \frac{dV}{dt} = E_L - V + R_m I \tag{3.1}$$

since the membrane time constant $\tau_m = c_m r_m$ and membrane resistance $R_m = \frac{r_m}{A}$. As shown in figure 3.1 the membrane integrates input current until some threshold V_Θ is reached, at which point the neuron spikes and the membrane potential is held in reset for some refractory period; in the absence of input, the charge across the membrane leaks until the equilibrium potential is reached. The LIF model thereby captures the essential dynamics of the Hodgkin and Huxley model, with much reduced computational complexity. The synaptic response to an afferent spike at time t_s is typically modelled as an exponential current of the form

$$\tau_s \frac{dI}{dt} = -I + w\delta(t - t_s) \tag{3.2}$$

where τ_s is the synaptic-current time constant, w is the peak current amplitude or *weight* of the synapse, and $\delta(x)$ is the discrete-time Dirac delta function, which returns 1 when $x = 0$ and 0 otherwise. Thus, the current from all spikes onto all synapses with a common time constant is given by

$$\tau_s \frac{dI}{dt} = -I + \sum_{i=0}^{n} w_i \sum_{j=0}^{m_i} \delta(t - t_{ij})$$

where t_{ij} is the time of the j^{th} spike onto the i^{th} synapse. Synaptic responses to spikes may also be modelled as any function of any model variable; common approaches include Dirac delta functions of currents and exponential functions of conductances.

The abstract nature of the LIF neuron allows model parameters to be derived analytically from cell electrophysiology with relative ease. The membrane resistance of

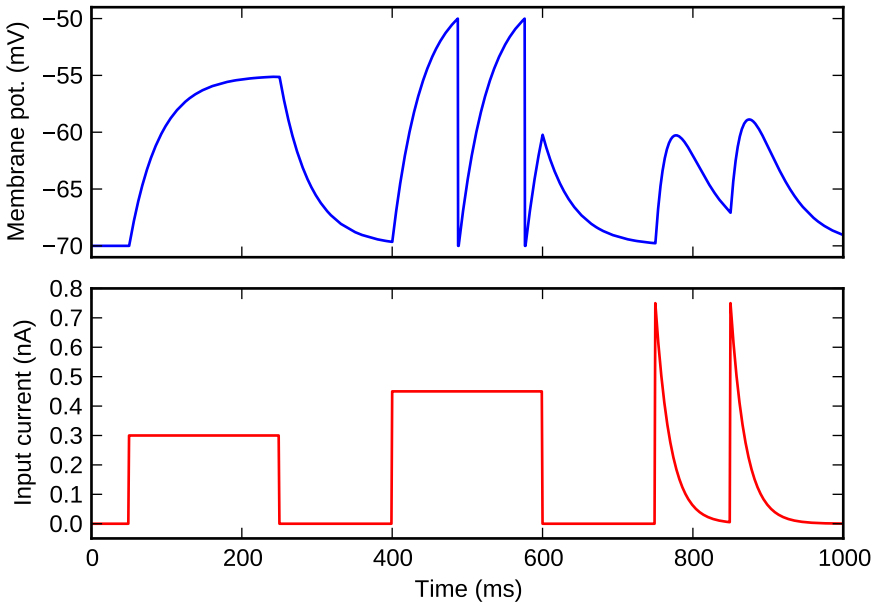


Figure 3.1: Current integration and leak in the LIF neuron model.

a neuron may be found by injecting a constant current into the cell and recording the membrane potential. Near the asymptote $dV/dt \approx 0$, so equation 3.1 may be rewritten

$$V(t) = E_L + R_m I$$

and rearranged to obtain R_m; the equilibrium potential may be likewise ascertained using a nil current. Also using constant current injection, the analytical form of the LIF model given by Dayan and Abbott (2001) can be rearranged to find the membrane time constant in terms of the asymptotic membrane potential and the observed membrane potential at some time t:

$$V(t) = E_L + R_m I - R_m I e^{-t/\tau_m}$$

$$\ln\big(V(t) - E_L - R_m I\big) = \ln\big(-R_m I\big) - \frac{t}{\tau_m}$$

$$\frac{t}{\tau_m} = \ln\left(\frac{-R_m I}{V(t) - E_L - R_m I}\right).$$

Electrophysiological data often describe observed postsynaptic potentials, such as those at $t = 750$ in figure 3.1. To reproduce these results in simulation, the weights and

time constants of the causative synaptic currents must be derived from these descriptions. The time constants of time-varying synaptic currents may be found from the analytical form of the LIF model given by Brunel and Sergi (1998)

$$V(t) = E_L + w\tau_s \frac{R_m}{\tau_m - \tau_s}\left(e^{-t/\tau_m} - e^{-t/\tau_s}\right) \tag{3.3}$$

which considers a single postsynaptic potential initiated at $t = 0$. The derivative of the equation at the postsynaptic potential peak, where $dV/dt = 0$, is

$$0 = w\tau_s \frac{R_m}{\tau_m - \tau_s}\left(\frac{e^{-t/\tau_m}}{\tau_m} - \frac{e^{-t/\tau_s}}{\tau_s}\right)$$

that can be rearranged to a form in which binary search may be used to find an appropriately precise value of τ_s

$$\frac{e^{-t/\tau_s}}{\tau_s} = \frac{e^{-t/\tau_m}}{\tau_m}.$$

Finally, this τ_s may be substituted into equation 3.3 to find the synaptic weight w, given an observed membrane potential at some time t.

The LIF model captures the essence of spike genesis, but it is insufficiently complex to express plausible subthreshold dynamics or phenomena such as bursting and spike-frequency adaptation. Izhikevich (2007) makes a dynamical-systems analysis of these activities, and so derives a computationally efficient model with a fast membrane potential variable v and a slow recovery variable u

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$

$$\frac{du}{dt} = a(bv - u)$$

$$\text{if } v \geq 30\text{mV}:\ v \leftarrow c,\ u \leftarrow u + d$$

where a, b, and d are dimensionless variables that control spike-frequency adaption and refractory period, and c is the post-spike reset value of the membrane potential. Varying these parameters results in a wide range of dynamics, as demonstrated with constant current input in figure 3.2, but the relationship between the biophysical properties of the neuron and a, b, c, and d is not readily apparent. For this reason, the Izhikevich model is somewhat more difficult to employ in biologically plausible simulations.

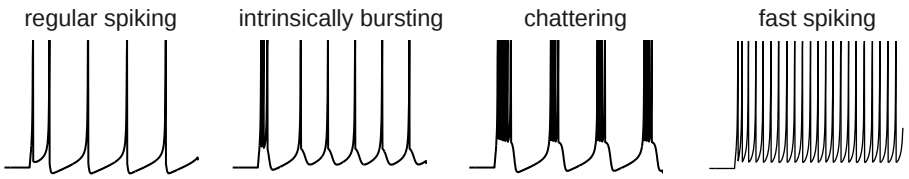


Figure 3.2: Four possible spiking patterns of the Izhikevich model.

3.2 Simulating many neurons and synapses

Neural computation appears to emerge from the parallel operation of many thousands or millions of neurons, which is extremely difficult to observe in any detail due to the scale and complexity of neural circuits. Simulation of neural tissue is a promising alternative methodology for elucidating the mechanisms of neural information processing. In this approach, neurons are simulated in networks derived from anatomical data, and the resultant activity is recorded in its entirety.

Most simulators employ an essentially common mode of operation: a network of neurons is specified by a user and compiled into simulation data structures; the simulation is executed in discrete time by one or more processors, each of which computes the state of one or more neurons based on intrinsic dynamics and external inputs; when neurons reach the membrane potential threshold, spikes are emitted and conveyed between processors; and, on completion, simulation data is returned to the user.

Neural information processing is characterised by massive computational parallelism, significant component heterogeneity, diffuse and specific communication patterns, and extreme energy efficiency (Atwell and Laughlin, 2001). The brain is a superlative computer that has yet to be fully replicated in silicon, and designers of neural simulators must be content with some trade-off in the achievable characteristics. Different simulation technologies therefore occupy particular regions of the design space, defined in terms of achievable parallelism, model flexibility and power consumption.

3.2.1 Software simulators

Software simulators are hardware-agnostic programs written in high-level languages, which are neither enhanced nor constrained by the executing computer. A great many software simulators exist, each with adaptations to different model types. The long-established NEURON simulator (Skrzypek, 1994; Hines and Carnevale, 2001; 2006) is primarily intended to model tens or hundreds of cells with complex morphologies at ion channel resolution, whereas Brian (Goodman and Brette, 2008) is orientated to-

wards simpler, point-neuron models in larger, thousand-cell networks. NEST (Gewaltig and Diesmann, 2007) is designed for yet greater model scale: the simulator comprises a front-end API for specifying models and a highly optimised, parallelised back end written in C++ for computing the states of tens of thousands of neurons (Plesser *et al.*, 2007). Many custom programs have also been written in C, C++ and MATLAB to accommodate model features that are not represented in the established simulators (Hines *et al.*, 2004). This multiplicity of technologies is somewhat unified by the simulator-independent PyNN language (Davison *et al.*, 2009) in which models may be specified for simulation on any platform that implements the PyNN programming interface.

Software simulators are cheap and easy to implement, obtain, maintain and modify. These programs are readily available by download from the authors' websites, and the source code or run-time environment is easily adaptable to new models that incorporate recent discoveries in neuroscience. Existing software simulators are almost certainly the appropriate platforms for quick deployment of small-scale models.

Models that comprise many millions of synapses, however, require considerable computational resources (Izhikevich and Edelman, 2008; Ananthanarayanan *et al.*, 2009). Such resources are only available in massively parallel computer architectures, which general-purpose hardware-agnostic simulators struggle to exploit efficiently. Parallel processing inevitably involves serial steps or interprocessor communication, and the proportion of a program taken up with these operations defines an asymptotic bound on the execution speedup that is possible with increasing numbers of processors (Amdahl, 1967). Therefore, parallel programs must be carefully tuned to minimise the time spent in the communication channels of a particular architecture. Both NEURON (Hines and Carnevale, 2007) and NEST (Brette *et al.*, 2007) have been adapted for parallel execution, but neither platform has yet been used for meganeuron simulations across many processors. This may be because of a fundamental mismatch between patterns of neural wiring and current interprocessor communication standards; the common MPI standard (Gabriel *et al.*, 2004) is designed to transmit kilo- or megabytes of data, and is therefore ill suited to conveying the binary pulses of action potentials. Ultimately, hardware-agnostic software simulators are useful platforms for small-scale and exploratory simulations, but may not scale to very large simulations of neural tissue.

3.2.2 Supercomputers

Simulations of the mammalian cortex have been conducted most successfully with custom software on conventional supercomputers. Markram (2006) describes the hardware

and software architecture of The Blue Brain Project, which intends to use 2^{17} processors in an IBM Blue Gene/L computer to simulate ten thousand cortical neurons and their fifty million synapses. Markram argues that detailed simulation of all neuron membrane conductances is necessary to produce a valid model of the cortex, against which Izhikevich (2007) argues that membrane potential dynamics may be accurately represented by a simple system of coupled differential equations. Izhikevich and Edelman (2008) exploit the efficiency of this approach to simulate a million-neuron, billion-synapse thalamocortical system on sixty processors of a Beowulf (Sterling *et al.*, 1995) cluster. Ananthanarayanan *et al.* (2009) use 2^{16} processors in an IBM Blue Gene/P machine to simulate a billion Izhikevich neurons and ten trillion synapses, and argue that this portends full-scale real-time simulations of the human cortex.

General-purpose supercomputers offer both model flexibility and significant parallelism, but may suffer from overwhelming power requirements. To assess the claim of Ananthanarayanan *et al.*, the power requirements of full-scale real-time simulations of the human cortex using current technology may be coarsely estimated as

$$P = P_m \cdot \frac{S_c}{S_m} \cdot T_m$$

where P_m is an estimate of the power drawn by the published work, S_c and S_m are the number of neurons in the cortex and the published model respectively and T_m is the number of real seconds taken to complete one second of simulation time. The latter term reflects a generous assumption that the time to compute a second of simulation time is inversely proportional to power expenditure (Salapura *et al.*, 2005).

Gara *et al.* (2005) report that the Blue Gene/L processor consumes ten watts during operation, and it is reasonable to assume that its successor, Blue Gene/P, has similar specifications. Ananthanarayanan *et al.* used 2^{16} such processors in simulation, suggesting a value $P_m = 10\text{W} \cdot 2^{16} \approx 655\text{kW}$. They simulated $1.6 \cdot 10^9$ of the $\approx 10^{10}$ neurons in the cortex (Braitenberg and Schüz, 1991, $\frac{S_c}{S_m} \approx 10$) at a rate of "643 seconds for one second of simulation per Hz of [spiking] activity" implying $T_m = 1{,}286$ at a plausible global average 2Hz neural spiking rate (Neymotin *et al.*, 2011). Consequently

$$P \approx 655\text{kW} \cdot 10 \cdot 1,286 \approx 8.4\text{GW}$$

which is orders of magnitude more power than a research institution may feasibly draw (Dongarra *et al.*, 2011). This gross estimate says nothing of the biologically fidelity of the simulation, but even the simplest spiking neuron models are no more than three

times more computationally efficient than the Izhikevich (2004) equations. Likewise, details of synapse models and dendritic compartments are overlooked, because they do not address the fundamental issue: the size of the power problem is such that the promises of Moore's law, which states that transistor size and power requirements decrease exponentially over time, do not permit power-feasible brain-scale simulations on conventional supercomputers before transistor dimensions reach atomic limits.

3.2.3 Field-programmable gate arrays

Customised processors may significantly outperform general-purpose processors in spiking neuron simulations. Such hardware may be prototyped in field-programmable gate arrays (FPGAs) that allow designs to be tested without the difficulty and cost of making application-specific integrated circuits (ASICs). Pearson *et al.* (2005), Rice *et al.* (2009) and Cassidy *et al.* (2011) implemented custom processors in FPGAs and the latter achieved real-time simulation of one million neurons. FPGAs may also be used as a reconfigurable substrate for arbitrary axonal wiring between processors, which would address the significant problem of routing the outputs from one simulated neuron to the inputs of another.

However, the typical connectivity degrees of neurons in biological networks are vastly greater than those of logic gates in silicon circuits, and FPGAs are not designed for such wiring densities. Furthermore, as Cassidy *et al.* admit, an ASIC implementation of any processor design will show significantly better computational performance and energy efficiency than its FPGA counterpart. Indeed, the reconfigurability of FPGAs is generally detrimental to their material cost, computational performance and power requirements. So, although they are a useful exploratory tool, successful prototypes should be fabricated as regular ASIC processors.

3.2.4 Graphics processing units

Simulations of neural tissue exhibit significant data parallelism, in that an identical stream of arithmetic instructions is used to compute the membrane potential of every neuron. Graphics processing units (GPUs; Fatahalian and Houston (2008) give an overview) currently contain tens or hundreds of arithmetic units that may execute a single instruction stream on many data elements simultaneously, thereby computing the state of many neurons in parallel. The capabilities of GPUs have generated significant interest in the neural modelling community (Nageswaran *et al.*, 2009; Bhuiyan *et al.*,

2010; Fidjeland and Shanahan, 2010; Han and Taha, 2010; Pallipuram *et al.*, 2011; Nere *et al.*, 2012) and almost all researchers report 10x to 1,000x improvement in simulation performance on GPUs over conventional central processing units (CPUs). However, evaluations of GPU performance typically suffer from methodological shortcomings and expose quietly reported limitations in the GPUs themselves.

Lee *et al.* (2010) assess the claim of fantastic GPU performance by running fourteen common scientific algorithms on a multicore CPU and a GPU. When each algorithm is carefully optimised for both platforms, the GPU performs on average only 2.5x better than the CPU, despite the fact that the former is more than three times more powerful in terms of data-parallel operations per second.

Memory bandwidth dictates the rate at which data may be read from or written to memory, and bandwidth limitations present a significant problem to GPUs due to the immense rate at which the numerous arithmetic units consume or produce data. Nageswaran *et al.* (2009) report that the memory bottleneck is the ultimate performance limitation of their work; Pallipuram *et al.* (2011) note that certain computations cannot be usefully performed on the GPU because of the cost of transferring data from the CPU; Bhuiyan *et al.* (2010) show that the examined GPU only outperforms comparable multicore processors when the number of instructions executed greatly exceeds the number of bytes transferred from or to memory; and Han and Taha (2010) find that a cluster of sixteen GPUs achieves only a $14\times$ speedup over a single GPU, presumably due to the cost of communicating spikes between neurons.

GPUs are cheap commodity platforms that show excellent performance on certain algorithms, particularly those with high ratios of computation to communication. However, simulation of neural tissue has the opposite property, so the parallelism available in GPUs cannot be applied to the task. Reports of GPUs' superlative performance beside CPUs either fail to control for differences in peak performance or omit the cost of data transfer between the two units.

3.2.5 Analogue circuits

Digital computers simulate neural dynamics by numerical approximation, but neurons can also be emulated by analogue integrated circuits that use subthreshold transistor states to mimic transmembrane ion channels (Mead, 1989; Indiveri *et al.*, 2011). Analogue circuits typically require few transistors and little power per neuron or synapse, and may compute many tens or thousands of simulated seconds in a single real second, greatly outperforming their digital counterparts in most regards. However, digital

computers are typically programmable, in that they execute instructions from a mutable memory, whereas the operation of analogue hardware is largely determined during fabrication. This may be a problem in the absence of a universal definition of neural dynamics because the design and tooling costs of integrated circuits are enormous, so the manufactured neurons must be correct according to both current *and future* understandings of neural operation.

This problem is partially addressed by neural models that can reproduce many known spiking activities (Brette and Gerstner, 2005) but the problem of spike transmission, which so frustrates FPGAs and GPUs, has typically limited analogue implementations to a handful of emulated neurons connected by a crossbar switch. The BrainScaleS project intends to address this problem by combining analogue circuitry for neural emulation with digital packet-switched routers for spike transmission (Schemmel *et al.*, 2008; 2010). As with most integrated circuits, some hundreds of chips will be manufactured on a single silicon wafer but, in radical contrast to the usual process of slicing the wafer and packaging each chip separately, a communications fabric will be deposited directly on the wafer to form a single package. Interwafer communication will be handled by a backplane to which multiple wafers may be connected. This approach offers large-scale simulations that run many times faster than real-time and consume little power, but also presents a problem in configuring arrays of analogue components that are exquisitely sensitive to both manufacturing and environmental conditions. The significant promise of the BrainScaleS hardware depends on the solution to this problem.

3.2.6 SpiNNaker

SpiNNaker is a computer architecture designed to simulate very many neurons and synapses in real-time. By emulating the structure and function of neural computation (Furber and Temple, 2006) the architecture is intended to address the Grand Challenge of understanding the brain and mind (Hoare and Milner, 2005). This has motivated design of a machine which contains up to 2^{16} multiprocessor chips, connected by a flexible asynchronous communications network (Plana *et al.*, 2007) that conveys simulated action potentials.

Processing

A SpiNNaker chip, as shown in figure 3.3, contains eighteen homogeneous processors: a *monitor* for administration, sixteen *application* processors for simulation, and one spare

to account for faults in the other seventeen. Each application processor is responsible for computing the dynamics of some number of neurons and their associated synapses, and has 32 kilobytes of instruction memory in which programs are stored and 64 kilobytes of data memory that contain program and neuron states. Synapse states and any other simulation data are stored in a 128-megabyte off-chip memory. Each processor has a communications controller through which neural spikes are communicated to and from the on-chip router, a direct memory access controller with which to read and write off-chip memory, and a timer peripheral that generates periodic signals to prompt computation of neuron states.

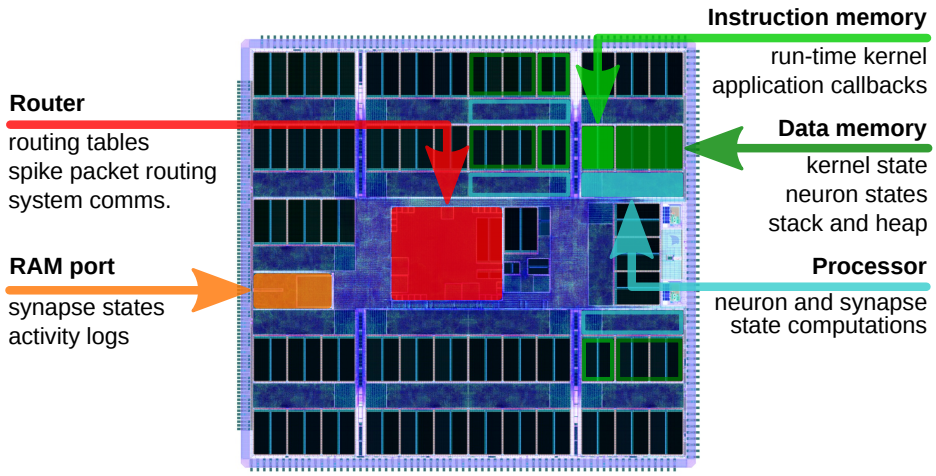


Figure 3.3: View of the SpiNNaker chip, labelled with key components.

SpiNNaker is an unusual high-performance architecture, in that it uses many low-performance processors to achieve great computational power. A microprocessor dissipates *static power* as a function of supply voltage and transistor leak current and *dynamic power* as a function of supply voltage and transistor switching (clock) frequency. High clock frequencies permit high computational throughput, but they also call for high dynamic power dissipation and are only achievable with low-threshold high-leakage transistors that suffer significant static power dissipation. Consequently, for workloads that may be efficiently shared, multiple low-performance processors may outperform a single high-performance processor in terms of both power consumption and computational throughput. For this reason, SpiNNaker uses ARM968 processors clocked at a relatively slow two hundred megahertz, and eschews complex integer division and floating-point arithmetic circuits to reduce static power dissipation.

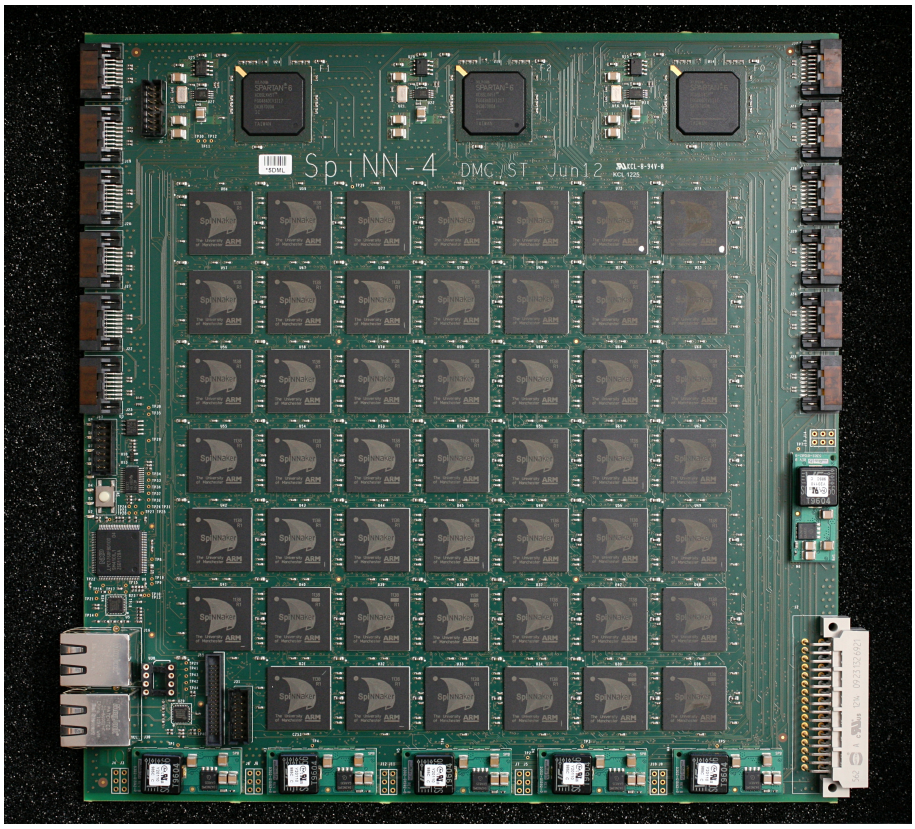


Figure 3.4: SpiNNaker production board.

Communication

Each chip contains a packet-switched router that forms links with all eighteen processors on-board and the routers of six neighbouring chips: North, South, East, West, North-East and South-West. The chips arranged on the 48-chip SpiNNaker board, shown in figure 3.4, thus form a plane through which packets may be communicated. Multiple production boards may also be connected through the board-to-board serial links, shown top-left and -right, and the expanded plane may be rolled into a toroid to ensure that the average interchip distance is constant throughout the machine. Ethernet ports, shown bottom-left, link the board to a host computer.

The on-chip routers may be programmed to emulate axons along which action potentials are transmitted. Procedurally: each neuron is uniquely identified by a 32-bit routing key; when a processor simulates an action potential, it sends the corresponding neuron's routing key to the on-chip router; the router compares this key against a routing table and delivers duplicate keys to one or more of the neighbouring routers or on-board processors; neighbouring routers repeat the process of look-up and delivery

according to their own routing tables; and, finally, each receiving processor induces synaptic currents in a subset of its own neurons selected according to the presynaptic neuron identified by the routing key. In this manner, SpiNNaker is specialised to communicate action potentials from any neuron to any subset of neurons in the machine. Routers are capable of processing packets at a rate which promises scalability to billion-neuron simulations (Navaridas *et al.*, 2009; 2010).

The routers also contain three further subsystems for maintenance: the nearest-neighbour system facilitates communication of 32- or 64-bit packets between the monitor processors of adjacent chips; the fixed-route system delivers marked packets to the nearest Ethernet port; and the point-to-point system, once configured, conveys packets comprising a 32-bit header and 32-bit payload between any pair of monitor processors, with the aid of dedicated routing tables in each chip.

3.3 Notable simulations

An enormous body of literature exists regarding the simulation of neural tissue. At the time of writing, the online ModelDB (Hines *et al.*, 2004) lists more than seven hundred examples and is by no means comprehensive.

Douglas *et al.* (1989) are amongst the first to simulate a significant volume of neural tissue, albeit at coarse granularity. They represented excitatory cells of the cortex as two firing-rate-coded units corresponding to cells in the supra- and infragranular layers, and represented inhibitory cells in all layers as one further unit. Such a network is capable of reproducing the *in vivo* activity of the visual cortex under thalamic stimulation. A paucity of anatomical data made the construction of more detailed models difficult, until a collaboration with Binzegger *et al.* (2004) produced the detailed map of cat visual cortex discussed in the previous chapter. Twenty years after the initial model, Binzegger *et al.* (2009) simulated cat visual cortex in unprecedented detail as nineteen populations of rate-coded leaky integrator neurons. Parameters were set in accordance with biological data: for a given synaptic current, inhibitory neurons were set to fire more readily than excitatory; the synaptic weights of the inhibitory projections were set to be stronger than those of the excitatory projections; and the majority of synapses in the model were distributed between excitatory populations, particularly in recurrent projections of the supragranular layer. The authors show that network dynamics are dominated by these supragranular projections and argue, in opposition to Bruno and Sakmann, that such recurrence amplifies weak thalamic signals to cortex. They suggest

that such amplification is selective of input features that match particular patterns of synaptic weights in the supragranular projections; thus, the cortex may fundamentally be a processor that selects, strengthens and relays preferred sensory signals.

The quantitative map of cat visual cortex produced by Binzegger *et al.* has precipitated numerous further simulations based upon the data. Haeusler and Maass (2006) treat the circuit as a binary classifier, and demonstrate that the laminar topology outperforms a structureless counterpart with otherwise identical statistical properties. Haeusler *et al.* (2009) extend this work to compare the performance of circuits based upon data from Binzegger *et al.* and Thomson and Lamy (2007). Wagatsuma *et al.* (2011) consider the attentional properties of the circuit and show that it is capable of reproducing biological phenomena of response to attended stimuli. Neymotin *et al.* (2011) demonstrate that biologically plausible oscillations may be induced in simulation of a small volume of cortex, and Izhikevich and Edelman (2008) demonstrate similar properties in a scaled-down model of the entire human cortex.

Izhikevich and Edelman present a model of the cortical plane comprising one million neurons, wired intracortically according to Binzegger *et al.* (2004). Intercortical connectivity was determined by original data on white-matter tracts generated by diffusion tensor imaging. Cells were simulated with detailed morphologies using the first author's simple model of spiking neurons (2003) and synapses were simulated with complex current and weight dynamics according to presynaptic cell type. Baseline activity was induced in the network using small artificial postsynaptic potentials, and the firing-rate power spectrum of each area was observed. The simulations show that synaptic-weight plasticity is a homeostatic mechanism that modifies the functional topology of the network, which then forms a substrate for the characteristic activity of the cortex. During a half-hour period of artificial stimulation, the network is adjusted under synaptic plasticity such that stable neural activity persists following the removal of the external drive. Furthermore, the power spectra of this activity correspond area-by-area to that observed in human electroencephalogram recordings, despite the structural homogeneity of the simulated cortical plane. This suggests that corticocortical signals through the white matter dynamically shape the function of each cortical area.

The visual cortex has been the focus of significant interest in biological and computational experiments (Hubel and Wiesel, 1962; Binzegger *et al.*, 2009) although it presents certain challenges to the investigation of cortical function. The retina of seeing animals has a great density of photoreceptors, so the amount of information transmitted to the cortex about a visual scene is large. Furthermore, visual stimuli are contin-

uous rather than discrete, visual information is preprocessed by the retina, and there is no clear mapping from retinal output to the cortical plane (da Costa and Martin, 2010). Consequently, it can be difficult to associate particular information in the sensory stimulus to particular neural activity in the cortex. In contrast, each whisker in the rodent somatosensory system is associated with exactly one barrel in the cortex, and whiskers may be perturbed by both discrete and continuous stimuli. For these reasons, Arabzadeh *et al.* (2004) are able to apply methods from information theory to precisely quantify the relationship between stimulus parameters and firing rates in the granular layer. This quantity is important if models of neural tissue are to be validated against their biological counterparts.

Kyriazi and Simons (1993) present exemplary simulations of thalamocortical response transformations in the rat barrel. As discussed in the previous chapter, Simons and Carvell (1989) hypothesise that sensitive inhibitory and nonlinear excitatory responses to synaptic currents enable cortical representations of whisker deflections that are not evident in the thalamic inputs. Kyriazi and Simons captured these principles in a one-hundred-cell model of barrel cortex, in which seventy neurons were excitatory and thirty were inhibitory. The output spikes of each neuron were generated probabilistically as a sigmoid function of membrane potential, and each spike was followed by a refractory period. Excitatory neurons were tuned to be highly nonlinear in their response to input, such that they spiked vigorously at high membrane potentials and were quiescent otherwise, and inhibitory neurons were tuned to be more broadly responsive, as shown on the left of figure 3.5. The barrel was stimulated with spike trains recorded directly from the rat thalamus during single whisker and adjacent-then-principal deflections. Upon carefully tuning thalamic and intracortical synaptic weights, the authors show that the simulated barrel accurately reproduces the strong on- and weak off-response of the rat barrel. More compellingly, they demonstrate that the same set of parameters may be used to reproduce the other three thalamocortical response transformations observed by Simons and Carvell, including cross-whisker response suppression. Kyriazi and Simons find that severing intracortical axons in the model undermines this response transformation, and so argue that inhibition is a key factor in determining, rather than simply regulating, cortical information processing.

The thalamus produces similar numbers of spikes in response to whisker deflection onset and offset, but the synchrony and peak rate of the spike trains vary. Pinto *et al.* (2003) review research on the stimulus parameters that are reflected in the cortical response, and present a novel dynamical-systems analysis of the rodent whisker barrel.

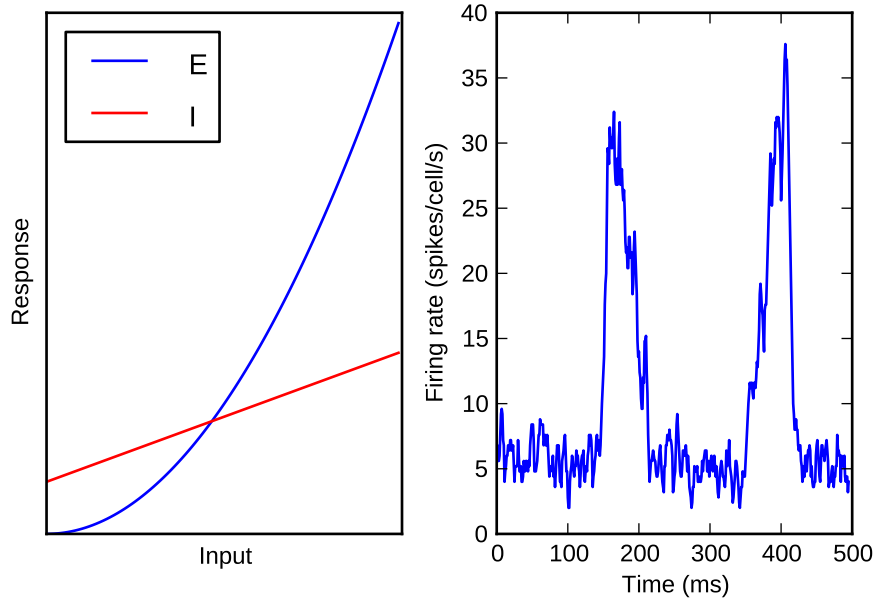


Figure 3.5: Example barrel response curves and thalamic stimuli.

Pinto *et al.* (1996) model the barrel using the same principles as Kyriazi and Simons, with the net firing rates of each population represented by a differential function of recurrent and extrinsic synaptic currents. The model was stimulated with a battery of spike trains with varying amplitudes and onset rates, as depicted on the right of figure 3.5, and the firing rates of the cortical populations were recorded. The results show that the model, like its predecessor, is capable of reproducing all four thalamocortical response transformations and that the response magnitude of the barrel is primarily determined by the onset rate, or synchrony, of the simulated thalamic stimuli. Pinto *et al.* (2000) corroborate these findings with animal experiments, which show that whisker deflection speed determines the spiking synchrony of the thalamus that in turn correlates with the level of firing in the barrel cortex. Finally, Pinto *et al.* (2003) apply dynamical-systems analysis to their 1996 model in order to explain the mechanisms underlying the thalamocortical response transformations. Figure 3.6 shows the phase plane with the firing rates of the excitatory and inhibitory populations on the x and y axes respectively, the excitatory nulcline in red, the inhibitory nulcline in blue, and the derivative of the excitatory firing rate in the background colour map. Increased thalamic drive moves the firing rate equilibrium away from the resting state: in the case of slow-onset stimulus, shown top, firing-rate derivatives remain small and the network state tracks closely with the moving equilibrium; in the case of fast-onset stimulus, shown bottom, the equilibrium moves rapidly from the resting state and the deriva-

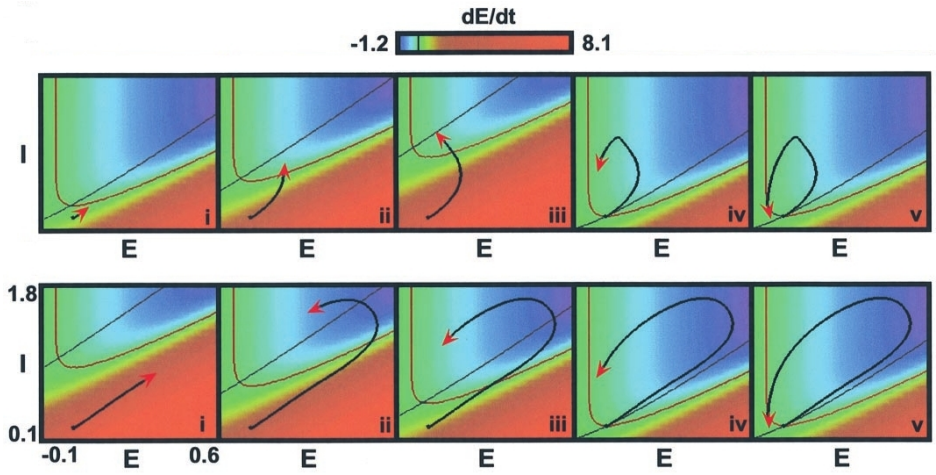


Figure 3.6: Phase plane analysis of barrel activity. Adapted from Pinto *et al.* (2003).

tives become large, so that the network state significantly overshoots the equilibrium and makes a wide orbit of the phase plane. The authors argue that this orbit is possible because of the strong supralinear response of excitatory neurons to synaptic input, but also note that the ultimate effect of intracortical synapses is an inhibitory damping of the response to weak or unsynchronised inputs. Since the synchrony of thalamic firing is proportional to the speed of whisker deflection, Pinto *et al.* argue that the rodent barrel cortex essentially acts to translate this quantity into a firing-rate code.

As Pinto *et al.* show, recurrently connected networks of neurons are dynamical systems, which may become hyper- or inactive depending on network parameters. Izhikevich and Edelman (2008) ensure stable, persistent network activity by a synaptic plasticity rule that modifies the functional topology of the model according to ongoing firing. Brunel (2000) approaches the problem of determining stable network activity analytically, by considering the balance of excitatory and inhibitory synaptic currents in a network. He simulates an externally driven, recurrently connected network of leaky integrate-and-fire neurons, of $0.8\times$ excitatory cells and $0.2\times$ inhibitory cells, in which every neuron receives a synapse from every other with equal probability. Excitatory synapses therefore outnumber inhibitory synapses four to one, and Brunel elegantly demonstrates that network activity depends on the ratio g of inhibitory to excitatory synaptic weights: when g is less than four, excitation dominates and the network is hyperactive; when g is greater than four, inhibition dominates and the network is practically inactive; and when g is approximately four a biologically plausible state of irregular, asynchronous, low-frequency firing is achieved. In this and other publications (1998; 1999; 2003) the author goes on to demonstrate various dynamical properties of

recurrent model architectures. This analytical description of network activity is particularly important, because many simulation studies achieve network stability through haphazard tuning of network parameters and such an approach calls into question the validity of the resulting model (Kyriazi and Simons, 1993; Haeusler *et al.*, 2009; Wagatsuma *et al.*, 2011; Neymotin *et al.*, 2011; Phoka *et al.*, 2012; Sharp *et al.*, 2012).

An exhaustive review of the simulation literature is clearly intractable, but a number of further studies are worthy of note: Maass and Markram (2006) present the *liquid state machine* as a possible model for computation in cortical circuits; Helmstaedter *et al.* (2007) review the motivations and techniques for simulating detailed models of individual cortical columns; Lang *et al.* (2011) actually do so, using data presented by Oberlaender *et al.* (2011); Johansson and Lansner (2007) argue that an abstract model of cortical columns may be used to simulate the entire human cortex; Symes and Wennekers (2009) examine the spatiotemporal dynamics of intercolumnar activity in a sheet of supragranular layer cells; Yger *et al.* (2011) further characterise the relationship between connectivity and activity in such a sheet; and Phoka *et al.* (2012) examine the changing response of a detailed barrel model with synaptic plasticity to successive stimuli.

3.4 Summary

Simulation of neural tissue appears to be a viable methodology for investigating brain function. Simple abstractions of neurons, such as the leaky integrate-and-fire and Izhikevich models, capture the essential electrophysiological properties of neural membranes and their signalling activities. Numerous technologies have been developed in order to simulate many such neurons, but all have significant limitations: portable general-purpose software simulators are unable to efficiently exploit the computational parallelism required for large-scale simulations; massively parallel machines consume too much power to be viable brain-modelling platforms; current experiments with GPUs and FPGAs do not demonstrate the necessary communications bandwidth to convey action potentials in intricate cortical networks; and current analogue hardware is difficult to program. Nevertheless, million-neuron simulations of the visual cortex have been achieved, and smaller simulations have produced concise theories regarding the function of the barrel cortex. The SpiNNaker project aims to advance the scale and detail of these simulations, using many low-power processors and neuromorphic communications hardware.

Chapter 4

The SpiNNaker Software Stack

Novel computer architectures require novel software stacks. The SpiNNaker architecture presents particular software challenges by virtue of its scale and communications infrastructure: boot software must configure a million-processor machine without the aid of dedicated command and control hardware; run-time software must offer a managed abstraction of hardware resources to user-defined computational tasks; and host-side software must translate neural tissue descriptions into machine-readable data structures for simulation. Ultimately, a SpiNNaker machine must be easily exploitable by researchers, without intricate knowledge of the architecture or the pitfalls of real-time parallel programming.

This chapter describes a novel software architecture for SpiNNaker. Section 4.1 reproduces the work of Sharp, Patterson and Furber regarding the boot-up and configuration procedures, presented at the 2011 International Joint Conference on Neural Networks. Section 4.2 explains event-driven computation on SpiNNaker and the associated application programming interface, drawing upon work presented by Sharp, Plana, Galluppi and Furber to the 2011 International Conference on Neural Information Processing. Section 4.3 briefly describes the PyNN interface to SpiNNaker through which models are specified from the anatomical literature, as presented by Galluppi, Davies, Rast, Sharp, Plana and Furber at the 2012 International Conference on Computing Frontiers, although the majority of this work must be credited to the first author.

4.1 Boot-up and configuration

SpiNNaker is a homogeneous mesh of chips, which at boot-up are undifferentiated by unique addresses and unable to probe their environment. Architectures such as

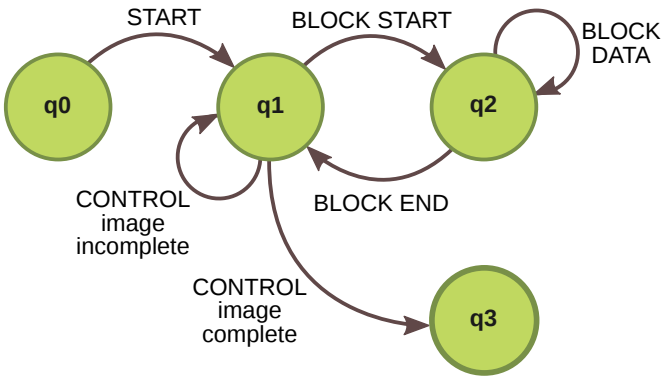


Figure 4.1: Flood-fill reception control.

the Blue Gene/L use dedicated hardware to configure machines (Haring *et al.*, 2005) but SpiNNaker eschews such systems in favour of topological flexibility and reduced complexity, so management must be done in software. This software must initialise hardware, load an operating system, set up an address map for the machine, and provide support for simulations (Khan *et al.*, 2009; Sharp *et al.*, 2011a; Patterson *et al.*, 2012).

4.1.1 Boot-up

Each processor in a chip is initially booted by a program stored in on-chip read-only memory (ROM). The ROM is immutable so program correctness is paramount, and for this reason the ROM code is limited in functionality to hardware testing and reception, but not retransmission, of more general software.

At power-on, each processor copies the ROM program into instruction memory and executes it. The program begins with tests on processor peripherals such as the communications and direct memory access (DMA) controllers; in the event of a failure, an error code is recorded and the processor disables itself. Processors that pass testing then race to access a system register that decides a *monitor* processor, which is thereon responsible for chip administration; the remaining cores become *application* processors, responsible for running simulations, and wait for messages from the monitor. Finally, the monitor tests chip peripherals such as the router and shared memories, disabling the chip in the event of failures, and enters a wait state.

A program is transmitted, by a mechanism discussed subsequently, to a chip via nearest-neighbour packets over the interchip links. Receipt of each packet raises an interrupt in the monitor, which calls a handling function according to the contents the packet key. Figure 4.1 shows the state machine that controls program loading. The process is initiated by a `START` packet that signals the number of blocks in which the

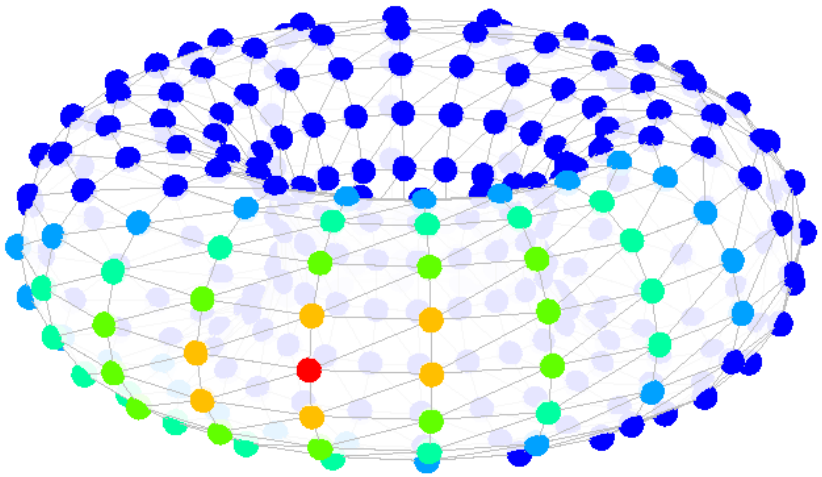


Figure 4.2: Representation of flood-fill in a toroidal topology.

impending program is to be transmitted. Subsequently, a `BLOCK_START` packet precedes one or more `BLOCK_DATA` packets, which each carry four bytes of program code that are stored by the monitor in shared RAM. A `BLOCK_END` packet concludes a block and carries a 32 bit cyclic redundancy checksum for validating the block; in response, the monitor uses the DMA engine to generate a new checksum for comparison, whilst copying the block from shared RAM to a data memory location determined by the block ID. Finally, following a number of blocks depending on the image size, a `CONTROL` message prompts the monitor to run a routine in ROM that copies the program binary from data memory to instruction memory and begin execution.

4.1.2 System configuration

Following boot-up, a small operating system is loaded into the machine. The operating system is responsible for self propagation through the machine, establishing chip addresses and communication channels, and loading simulation programs and data.

Propagation

The operating system is first transmitted to an Ethernet-connected monitor processor on SpiNNaker using a block-wise protocol similar to the one described above, with trivial adaptations to the medium. To distribute the program to the rest of the machine, the monitor immediately retransmits the instruction memory contents in all directions over

the nearest-neighbour fabric, thereby fulfilling the role of the anonymous transmitter in the previous section. This *flood-fill* process, repeated iteratively by the monitor processor of each chip, is the mechanism by which system software is distributed from the host to all monitors of a SpiNNaker machine.

Figure 4.2 illustrates flood-fill commencing from an Ethernet-connect seed chip, depicted in red, and proceeding across the nearest-neighbour network to a distance of four links. The flood-fill wavefront in this toroidal topology touches upon $6d$ chips, where d is the number of links traversed. Consequently a flood-fill extending across n links from the seed affects $\sum_{i=0}^{n} 6i = 6\frac{n(n+1)}{2}$ chips. Assuming that transmission time across each link is invariant in the size of the wavefront, the time required for flood-fill is a root term of the number of chips in the machine. This suggests that flood-fill is an efficient mechanism for distribution of homogeneous binaries across large-scale SpiNNaker machines, as demonstrated in chapter 5.

Machine mapping

The toroidal topology of a SpiNNaker machine allows each chip to be uniquely identified by a two-dimensional coordinate. To establish a machine map, an arbitrary Ethernet-connected chip is assigned the origin address $(0, 0)$ and is informed of the selection by the host machine. The monitor processor of the origin chip computes its neighbours' relative addresses (the northern neighbour, for example, is $(x, y + 1)$ and the southwestern neighbour is $(x-1, y-1)$ all modulo the dimensions of the machine) and informs them of these coordinates via nearest-neighbour packets. Each monitor repeats this process until every chip has received an address.

With an address map established, it is then possible to configure the point-to-point routing system that allows packets to be sent between arbitrary pairs of monitor processors. To do so, an analog of the distance-vector routing protocol is employed. Briefly described: each monitor creates a hop-count table hc with entries for every other monitor initialised to infinity; at regular intervals every monitor transmits a nearest-neighbour packet, containing in the payload the source address (x_0, y_0) and a hop-count $h = 1$, to the set of neighbours $C(d = 1) = \{(x, y) \mid \text{hops}((x_0, y_0), (x, y)) = d\}$; each receiving monitor processor in $C(d = 1)$ checks if $h < hc((x_0, y_0))$, that is, if the packet represents a new shortest route to the source, and if so sets the point-to-point route to (x_0, y_0) to be the link upon which the nearest-neighbour packet arrived; and finally, upon the same condition, each monitor in $C(d = 1)$ increments h and retransmits the packet to chips in $C(d = 2)$, which repeat the process. By this mechanism, the

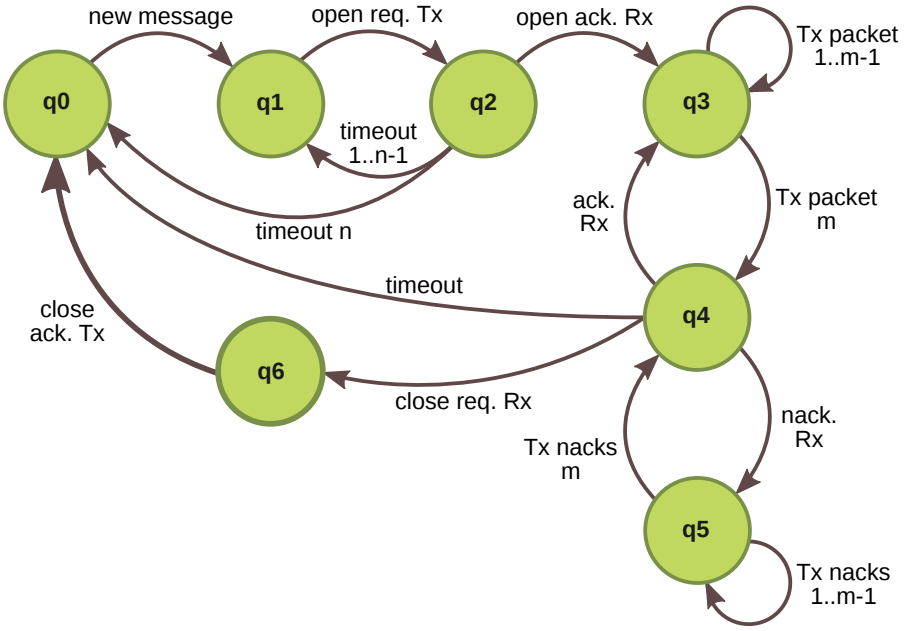


Figure 4.3: SDP transmission control.

shortest possible route between any pair of chips is found, in a process that is robust to permanently- or transiently broken interchip links.

Control and communications

The SpiNNaker datagram protocol (SDP) provides a control and communications infrastructure that is absent from the hardware. SDP messages consist of source- and destination-processor addresses, a command field, three arguments, and a variably sized payload, which may collectively be used to upload programs and data, trigger execution, and download results. In communication between the host machine and the Ethernet-connected monitor processor, these datagrams may be carried in the payload of a UDP packet; for transmission between the host and distal chips, the datagrams must be broken up and transmitted over the point-to-point network. This requires that both the transmitter and receiver maintain some state in software regarding the datagram, and that each transmitted packet includes some metadata with which the state is modified. Figure 4.3 shows the state machine of the software protocol used to transmit SDP messages as a series of point-to-point messages: the transmitter opens a connection with the receiver, transmits the m packets of the datagram with ID metadata, and waits for the receiver to acknowledge those received; any packets not acknowledged by the receiver are retransmitted, and when receipt is finally confirmed the connection is closed. The receiver implements a complementary state machine, and in both cases

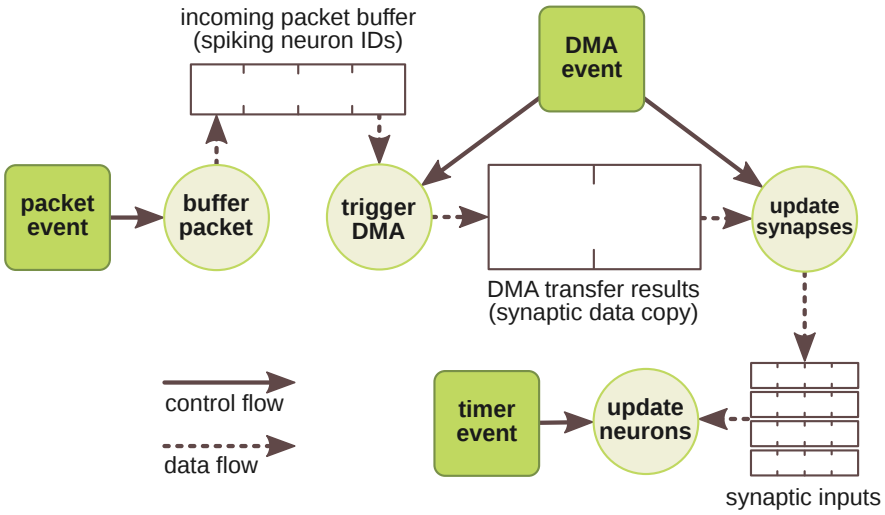


Figure 4.4: Events and corresponding tasks in a typical neural simulation.

an unexpected delay in receipt of a packet may cause the connection to be terminated. Finally, on receipt of a complete SDP message, the receiver uses the command and argument fields to determine the operation, such as program execution or fulfilment of a memory-load request, that should be performed on the payload.

With the operating system loaded, the machine configured, and the control and communications infrastructure set up, SpiNNaker is ready to execute simulations.

4.2 Event-driven computation

SpiNNaker simulations are *event-driven* in that all computational *tasks* follow from events in hardware, as shown in figure 4.4 (Sharp *et al.*, 2011b). On each processor, neuron states are computed in timesteps initiated by a local *timer event* with a programmable period; at each timestep, usually a millisecond in length, processors evaluate the membrane potentials of all of their neurons given prior synaptic inputs and deliver a packet to the router for each neuron that spikes. Spike packets are routed to all processors that model neurons efferent to the spiking neuron. Receipt raises a *packet event* that prompts the efferent processor to retrieve the appropriate synaptic weights from off-chip RAM using a background direct memory access (DMA) transfer. The processor is then free to perform other computations during the DMA transfer and is notified of its completion by a *DMA event*, which prompts calculation of the synaptic inputs to subsequent membrane potential evaluations. Should additional packets arrive while a DMA transfer is in progress, they are placed into a software buffer and are con-

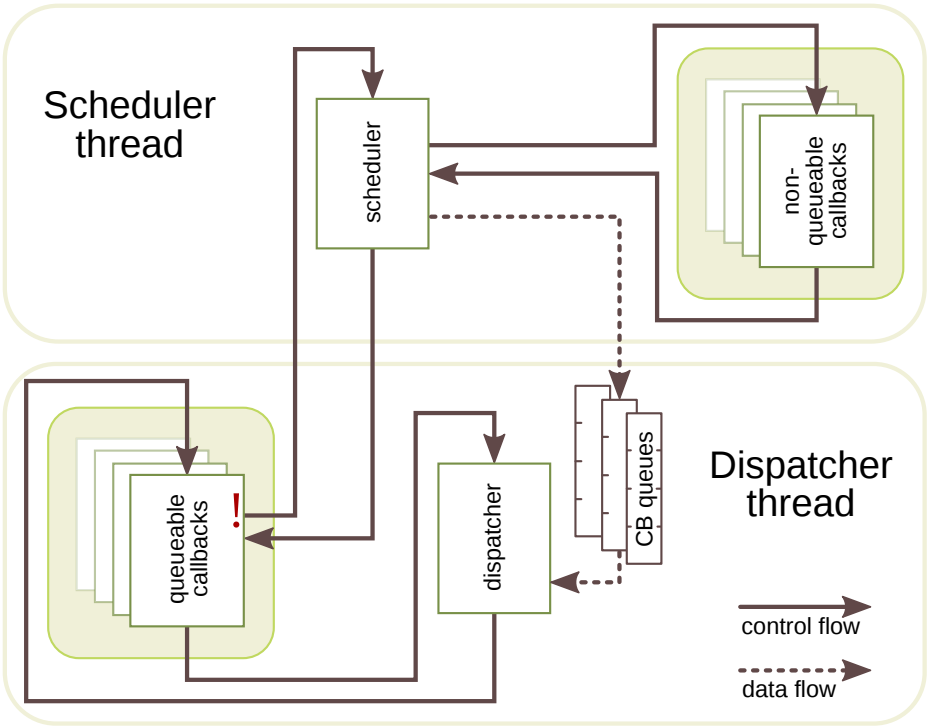


Figure 4.5: Control and data flow between the scheduler and dispatcher threads.

sumed one-by-one as each prior DMA completes. In the absence of events, processors go into a passive wait state to conserve power. It should be noted that, in this context, *events* should not be confused with those discussed by Brette (2006) and Hanuschkin *et al.* (2010) in the solution of membrane potential equations.

4.2.1 The ARK

Each application processor executes an instance of the application run-time kernel (ARK) which is responsible for providing computational resources to the tasks arising from events. Figure 4.5 shows the two ARK threads of execution that share processor time: following events, control of the processor is given to the *scheduler* thread that queues tasks; upon its completion, the scheduler returns control to the *dispatcher* thread that dequeues tasks and executes them. In terms of figure 4.4 a timer event schedules a neuron update task that is dispatched upon returning from the scheduler.

Tasks have *priorities* that dictate the order in which they are executed by the dispatcher. The scheduler places each task at the end of the queue corresponding to its priority and the dispatcher continually executes tasks from the highest-priority nonempty queue. Priority zero tasks are *nonqueueable* and are executed by the scheduler directly,

precluding any further scheduling or dispatching until the task is complete; priority minus one tasks are *preemptive* and are executed in a dedicated processor mode, which disregards the scheduler and interrupts all other operations until complete.

The ARK also records certain data about the execution of a program. Packet counters are read from the router hardware and stored in a diagnostic block, along with software counts of task and DMA queue overflows. At the end of a simulation, a user-level program built on the application programming interface may read the diagnostic block and append any further recordings, before offering the results to the host.

4.2.2 The API

The SpiNNaker application programming interface (API) allows a user to specify the tasks that are executed following an event. A user may write *callback* functions in C that encode the desired tasks, and register them with the ARK against particular events. At this point, it is illustrative of the API to derive a program that simulates the leaky integrate-and-fire (LIF) neuron with exponential-current synapses.

As discussed in the previous chapter, the subthreshold membrane potential $V(t)$ of the LIF neuron is determined by

$$V(t) = E_L + R_m I - (E_L + R_m I)e^{-t/\tau_m}$$

where τ_m is the membrane time constant, E_L is the equilibrium potential, R_m is the membrane resistance and I is a time-invariant input current. On exceeding the threshold V_Θ the membrane potential is set to V_{reset} and held there for some refractory period $refrac$. Where time-varying excitatory and inhibitory currents I_e, I_i occur the LIF neuron is typically presented as a dynamical system

$$\tau_m \frac{dV}{dt} = E_L - V + R_m\big(I_e - I_i\big)$$

in which a presynaptic spike induces an instantaneous rise of w_i (the *weight* of the synapse between the pre- and postsynaptic neurons) in a synaptic current that then decays exponentially according to the time constant τ_I

$$\tau_I \frac{dI}{dt} = -I + \sum_{i=0}^{n} w_i \sum_{j=0}^{m_i} \delta(t - t_{ij})$$

Here, I stands for the excitatory or inhibitory current, t_{ij} is the time of the j^{th} spike on

the i^{th} synapse, and δ is the Dirac delta function that is nonzero only when $t = t_{ij}$.

Jin *et al.* (2008) propose a methodology for solution of the Izhikevich (2003) equations on SpiNNaker, which generalises to arbitrary neuron models. Following this, the LIF equations are solved at millisecond intervals according to the difference equations

$$V(t+1) = V(t) + \lambda_m\Big(E_L - V(t) + R_m\big(I_e(t) - I_e(t)\big)\Big) \tag{4.1}$$

$$I(t+1) = I(t) + \lambda_I\big(-I(t) + \sum_{i=0}^{n} w_i \sum_{j=0}^{m} \delta(t - t_{ij})\big) \tag{4.2}$$

where $\lambda = 1/\tau$. $V(t)$ is computed exclusively by the timer callback; $I(t)$ is incremented by the DMA callback in response to each afferent spike and is decayed by the timer callback in response to each timer tick. Thus, the cost of modelling exponential-current synapses is constant per neuron, rather than a function of the number of input spikes.

Real numbers are represented in SpiNNaker as 32-bit values with a decimal point fixed, by software convention, after the 16th bit; this affords an approximate range and precision of $\pm 10^4$ and $\frac{1}{10^4}$ respectively, which tentative simulations suggest is minimally necessary for accurate neuron-state evaluation. Numbers are therefore represented in the machine as the product of the value x and a scaling factor $P = 2^{16}$, and multiplication of two such numbers must be followed by a right-shift to maintain the correct scale of the result. In C code, the operation $Pz = \frac{PxPy}{P}$ is implemented

```
// Given int x, int y
long long temp = x * y;
int z = (int) (temp >> LOG_P);
```

Division by numbers other than powers of two must be avoided due to the lack of hardware support for the operation in the ARM968 processor. Fortunately, it is possible to precompute the reciprocals of the divisors in the LIF model and substitute multiplication for division, as in equations 4.1 and 4.2.

Listings 4.1 and 4.2 show the SpiNNaker implementation of the LIF neuron (edited for clarity) upon which subsequent experiments are based. The main function of the application configures the hardware and registers callbacks for each of the tasks shown in figure 4.4 and a timer callback function computes the LIF equations. Omitted here for brevity, the packet callback receives input spikes and initiates DMA transfers of synaptic weights into local memory, and the DMA callback translates these weights into input currents to their respective neurons.

```
int main() {
  // Set time period, load sim. data and configure run-time code
  spin1_set_timer_tick(1000); // microseconds
  load_simulation_data();
  configure_simulation_environment();

  // Register callbacks using an API function and start simulation
  spin1_callback_on(PACKET_EVENT, packet_callback, PRIORITY_1);
  spin1_callback_on(DMA_DONE_EVENT, dma_done_callback, PRIORITY_2);
  spin1_callback_on(TIMER_EVENT, timer_callback, PRIORITY_3);
  spin1_start();
}
```

Listing 4.1: The main function of the SpiNNaker LIF implementation.

```
void timer_callback(uint ticks, uint null) {
  long long psc_e, psc_i, psc, psp, dv, di;

  // Iterate over each neuron in the processor
  for(uint i = 0; i < num_neurons; i++) {
    // Handle the spiking condition
    if(neuron[i].v >= neuron[i].v_thresh) {
      uint key = spin1_get_chip_id() << 16 | spin1_get_core_id() << 11 | i;
      spin1_send_mc_packet(key, NULL, NO_PAYLOAD);
      neuron[i].v = neuron[i].v_reset;
      neuron[i].refrac_clock = neuron[i].refrac_tau;
    }

    // Compute the membrane potential, if not in a refractory period
    if(neuron[i].refrac_clock == 0) {
      psc_e = neuron[i].psc_e[ticks];
      psc_i = neuron[i].psc_i[ticks];
      psp = (psc_e - psc_i) * neuron[i].v_resistance;
      psp = psp >> LOG_P;
      dv = neuron[i].v_decay * (neuron[i].v_rest - neuron[i].v + psp);
      dv = dv >> LOG_P;
      neuron[i].v = (int) (neuron[i].v + dv);
    }
    // Else decrement the refractory clock
    else {
      neuron[i].refrac_clock--;
    }

    // Compute synaptic-current decay (inhibitory current omitted)
    di = psc_e * neuron[i].exci_decay;
    di = di >> LOG_P;
    neuron[i].psc_e[ticks + 1] += (int) (psc_e - di);
  }
}
```

Listing 4.2: The timer callback of the SpiNNaker LIF neuron implementation.

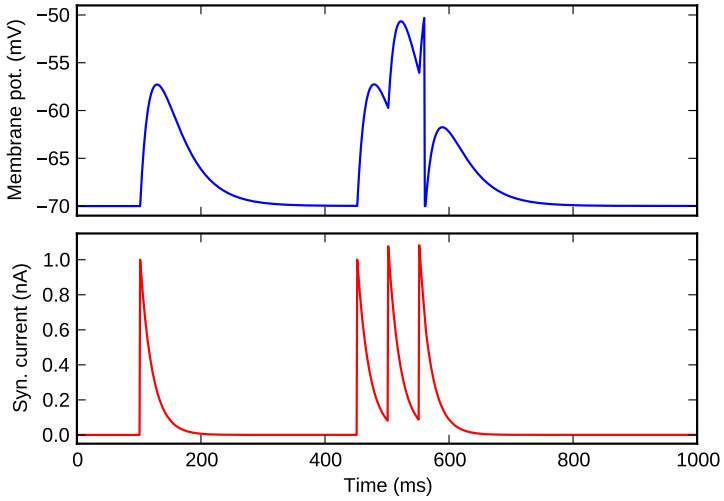


Figure 4.6: Simulation of a single LIF neuron.

Figure 4.6 shows a trace from a single LIF neuron, as simulated on a SpiNNaker chip and recorded in the corresponding off-chip memory. Each presynaptic spike causes an instantaneous rise and exponential decay in postsynaptic current (bottom) that is integrated by the membrane (top) such that the fourth spike raises the membrane potential to the threshold and elicits an output spike.

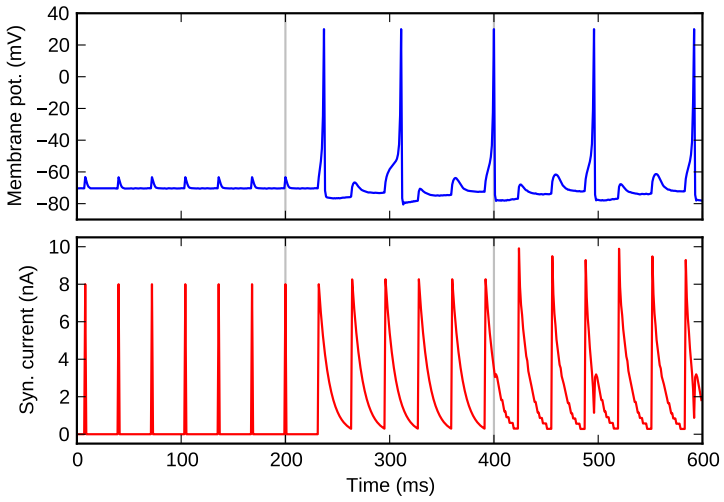


Figure 4.7: Simulation of a single Izhikevich neuron, with varying synapse models.

The API, however, allows arbitrary neuron and synapse dynamics to be simulated. Any model may be encoded in standard sequential C code, as in listing 4.2, and registered against the timer event. Figure 4.7 traces an Izhikevich neuron stimulated by delta-current, exponential-current, and exponential-conductance synapse models: the main function of this simulation registers the timer callback for the first model with the

ARK; after two hundred milliseconds, the first callback reregisters the second by the same `spin1_callback_on` function call, resulting in a change in membrane potential dynamics; after a further two hundred milliseconds, the second callback registers the third, causing another, slighter, variation. This is an unusual usage of the SpiNNaker application programming interface, but it serves to demonstrate the great flexibility of the software, which is of paramount importance in the exploratory research of neural simulations (Furber and Brown, 2009).

4.3 PyNN and PACMAN

SpiNNaker is designed to simulate many millions of neurons and billions of synapses. The PyNN application programming interface (Davison *et al.*, 2009) is used to specify these components, and the SpiNNaker partitioning and configuration manager (PACMAN) translates such specifications into machine-orientated data structures. PyNN allows researchers to build models from a library of components, currently including the leaky integrate-and-fire and Izhikevich neurons in the SpiNNaker implementation, without concern for the underlying simulation technology. Consequently, models written in PyNN are portable across simulators, which is particularly useful for verifying the results obtained from SpiNNaker against those of established platforms.

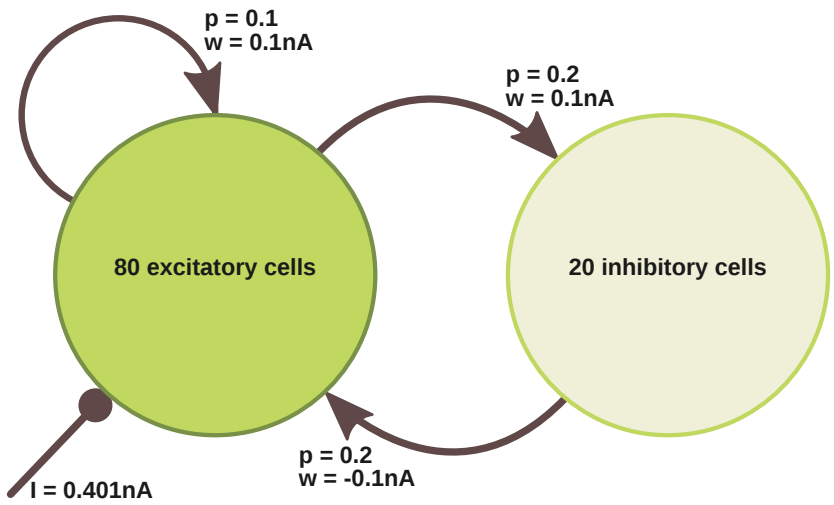


Figure 4.8: Simple network with recurrent projections.

Simulations are specified as *populations* of homogeneous neurons and *projections* of synapses with homogeneous dynamics and variable weights. The simple network of excitatory and inhibitory neurons in figure 4.8, for example, is built by the Python program in listing 4.3: firstly, the simulator module is imported and initialised; secondly, two populations are created with the parameters specified by the `lif` dictionary, and neurons of the excitatory population are set to receive an input current; thirdly, projections are built, such that a synapse with weight `w` is created with probability `p` between each pair of cells in the source and target populations; and finally, the simulation is run, and the results are retrieved and plotted (code not shown) as in figure 4.9.

```
# Import the SpiNNaker PyNN implementation and initialise the simulator
import pyNN.spinnaker as pynn
pynn.setup()

# Declare LIF parameters, populations, and input current
lif = {"v_rest":-70.0, "v_reset":-70.0, "v_thresh":-50.0, # mV
       "tau_m":40.0, "tau_syn_E":20.0, "tau_syn_I":20.0,  # ms
       "cm":40.0/50.0, "tau_refrac":1.0, "i_offset":0.0}  # nF, ms, nA
exci = pynn.Population(80, pynn.IF_curr_exp, lif)
inhi = pynn.Population(20, pynn.IF_curr_exp, lif)
exci.set("i_offset", .401)

# Declare projections
e_e_conn = pynn.FixedProbabilityConnector(p=.1, w= .1)
e_i_conn = pynn.FixedProbabilityConnector(p=.2, w= .1)
i_e_conn = pynn.FixedProbabilityConnector(p=.2, w=-.1)
e_e_proj = pynn.Projection(exci, exci, e_e_conn, target="excitatory")
e_i_proj = pynn.Projection(exci, inhi, e_i_conn, target="excitatory")
i_e_proj = pynn.Projection(inhi, exci, i_e_conn, target="inhibitory")

# Enable spike recording, run the simulation and retrieve the results
exci.record()
inhi.record()
pynn.run(1000)
esp = exci.getSpikes()
isp = inhi.getSpikes()
```

Listing 4.3: A PyNN specification of a simple network.

PACMAN implements the PyNN API in the form of the `pyNN.spinnaker` module, and thereby records references to each population and projection that is instantiated when a PyNN program is executed. On a call to `pynn.run()`, PACMAN maps the simulation entities to hardware: firstly, populations are split into subpopulations according to size and neuron model complexity, and the subpopulations are allocated to processors; secondly, routes for the specified projections are found between the al-

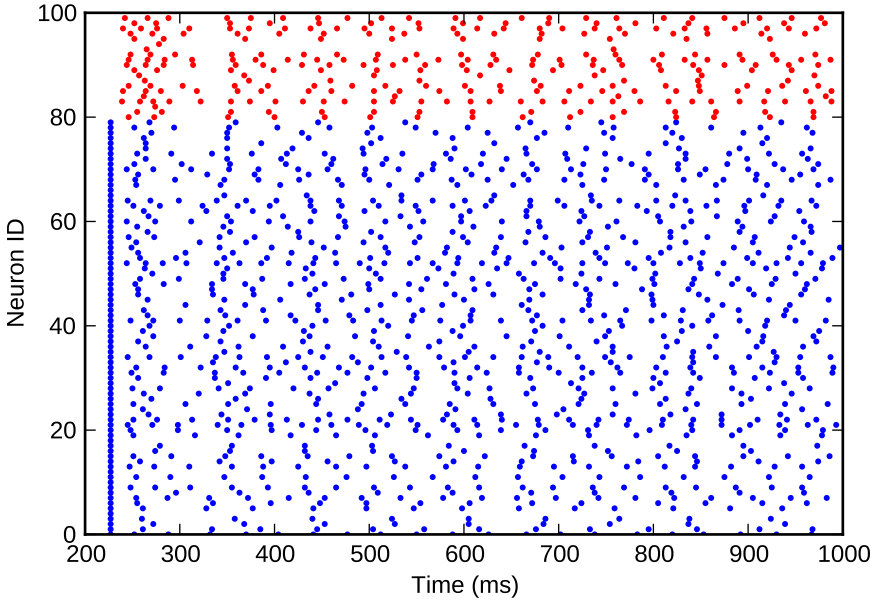


Figure 4.9: Spiking activity of the simple network in figure 4.8.

located processors; thirdly, data structures representing neurons, routes and synapses are built and uploaded to the target machine; and finally, the simulation is triggered. Following a simulation, recorded data may be retrieved with `Population.get_v()` and `Population.get_spikes()`, and `pynn.get_diagnostics()` may be called to retrieve the diagnostic data recorded by the API. Jin *et al.* (2010), Galluppi *et al.* (2010; 2012) and Davies *et al.* (2012) describe successive revisions of PACMAN, which follow broadly the same structure, in greater detail. However, for the purposes of this thesis, a small, custom implementation of PACMAN was written that deviates in some respects from the cited works for the purposes of flexibility and performance.

4.4 Testing

SpiNNaker depends upon a large software stack that spans both the host machine and the SpiNNaker chips. To ensure consistent correctness of simulations during further software development, a set of regression tests were developed. These tests execute deterministic simulations of increasing complexity, compare the results against against reference data, and report any discrepancies. Although it is impossible to declare the absence of all bugs, a baseline level of correctness is guaranteed by running the regression tests and checking the results after each modification to the software. To a limited

extent, this ensures that interesting simulation results are not simply the artifacts of programming errors. Figure 4.10 shows correct execution of the first six regression tests, which demonstrate from left to right and top to bottom: superimposed membrane potential traces of one hundred LIF neurons on each processor of a machine, all receiving some constant input current; postsynaptic potential traces of seventy five neurons distributed across three processors, all receiving periodic input spikes from another simulated neuron but with incremental synaptic weights; stacked postsynaptic potential traces of one LIF neuron on each processor of a machine, all receiving periodic input spikes from another simulated neuron; spike times of one hundred neurons on each processor of a machine (results from only one processor shown) all receiving some periodic input spikes from another simulated neuron; spike times of one hundred neurons split across two processors, receiving input spikes from an artificial spike source; and spike times of a Vogels-Abbott network benchmark specified by Brette *et al.* (2007).

4.5 Summary

Novel, massively parallel computer architectures are typically difficult to configure and program, especially for nonexpert users. SpiNNaker is designed for computational neuroscientists wishing to model large, complex volumes of neural tissue, so it is exceptionally important that the event-driven, parallel nature of the hardware is disguised. The presented software stack does exactly this: the event-driven programming interface supports arbitrary neural dynamics written in standard sequential C code, and the PyNN interface allows models to be specified in terms of neural populations and synaptic projections with no knowledge of the underlying hardware or software architecture whatsoever. As such, the significant power of a massively parallel SpiNNaker machine is made available to a wide audience of researchers.

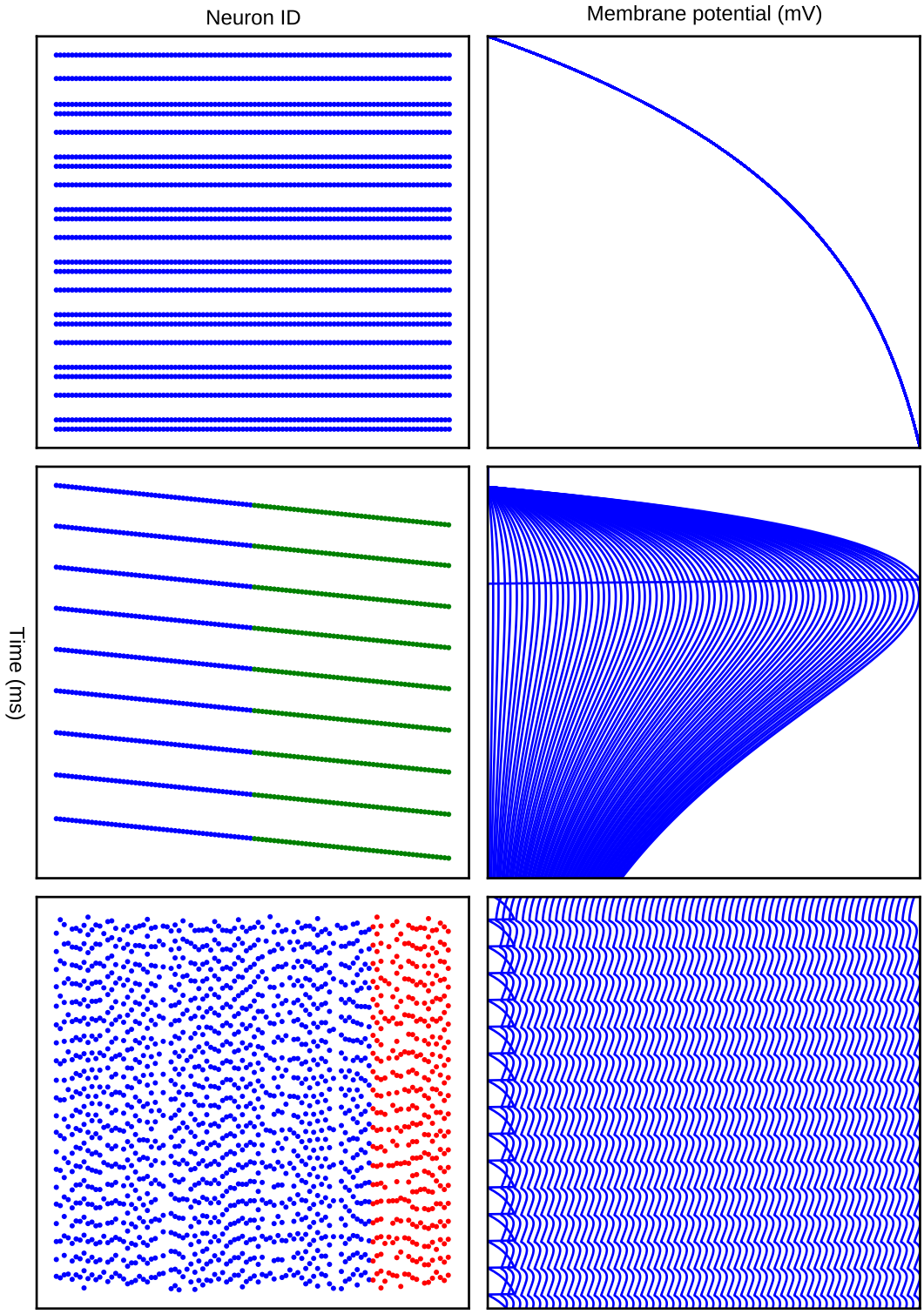


Figure 4.10: Simulation traces from regression tests.

Chapter 5

SpiNNaker Performance

SpiNNaker is expected to simulate large-scale models correctly, quickly and efficiently. To do so, the software must accurately evaluate neurons and synapses within the real-time constraints imposed by the hardware timer tick, and each chip must operate within its share of the energy budget of a full-scale machine. A series of experiments were conducted to sketch performance and power profiles of the software and hardware, and to compare results obtained on SpiNNaker with those from reference platforms.

This chapter demonstrates that SpiNNaker satisfies the design criteria regarding performance, power and correctness of simulation results. Section 5.1 reproduces a small contribution to the work of Patterson, Garside, Painkras, Temple, Plana, Navaridas, Sharp and Furber (2012) in *Journal of Parallel and Distributed Computing*, which shows that the low-level boot and configuration software is suitable for managing a full-scale, million-processor machine. Section 5.2 presents simulations of increasing complexity and shows for each that SpiNNaker accurately reproduces the output of reference simulators, while section 5.3 shows that SpiNNaker outperforms the reference platforms and presents a detailed execution profile; these data comprise an accepted submission by Sharp and Furber to the 2013 International Joint Conference on Neural Networks. Section 5.4 demonstrates the superlative power efficiency of SpiNNaker using detailed simulations of the cat visual cortex, which have been previously published by Sharp, Galluppi, Rast and Furber (2012) in *Journal of Neuroscience Methods*.

5.1 Low-level software behaviour

A series of experiments were conducted on SpiNNaker test chips, containing just two processors, to verify the correctness and performance of the boot ROM and operat-

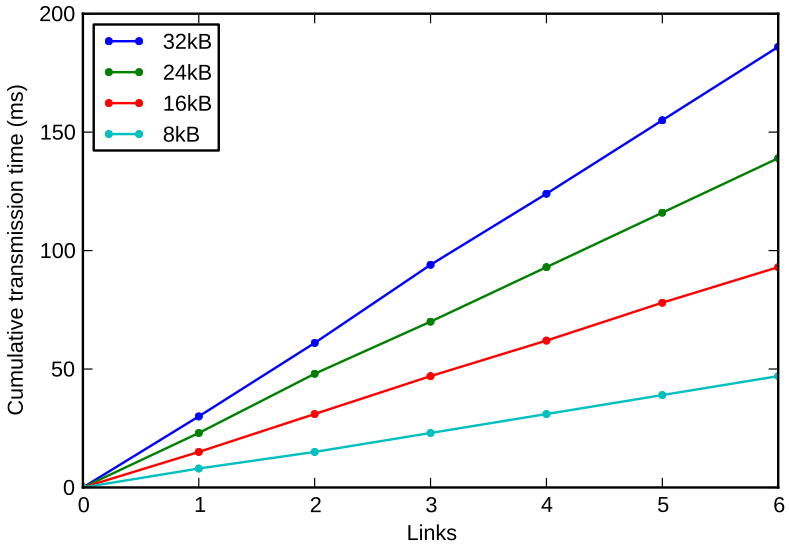


Figure 5.1: Flood-fill times along a chain of chips.

ing system protocols. The experiments aimed to determine that a large-scale SpiNNaker machine could be booted by flood-fill in tractable time, and that data could be transferred to and from a running machine likewise. These properties of the low-level software are important if the high performance of SpiNNaker is to be exploited: the advantage of a real-time simulator is negated if the configuration time outweighs that of computation. In these simulations, the processors were run at 150 megahertz and the the router, system bus and off-chip memory were run at 100 megahertz.

5.1.1 Flood-fill time

A number of four-chip test boards were arranged to create a chain of interchip links spanning seven chips, and an Ethernet connection was made between the *proximal* chip and the host. An oscilloscope probe was placed on one general-purpose input-output pin of each chip. A minimal version of the operating system was compiled that contained routines to switch the GPIO pin on start-up, functions for self propagation according to the protocol described in section 4.1.2, and varying amounts of padding. The host machine seeded this version of the operating system into the proximal chip, and the program then self propagated, link-by-link, to the distal chips. The time taken for propagation across each link was recorded.

Figure 5.1 shows propagation time for varying program sizes over varying numbers of links from the proximal chip. A 32 kilobyte program, the largest that can be accom-

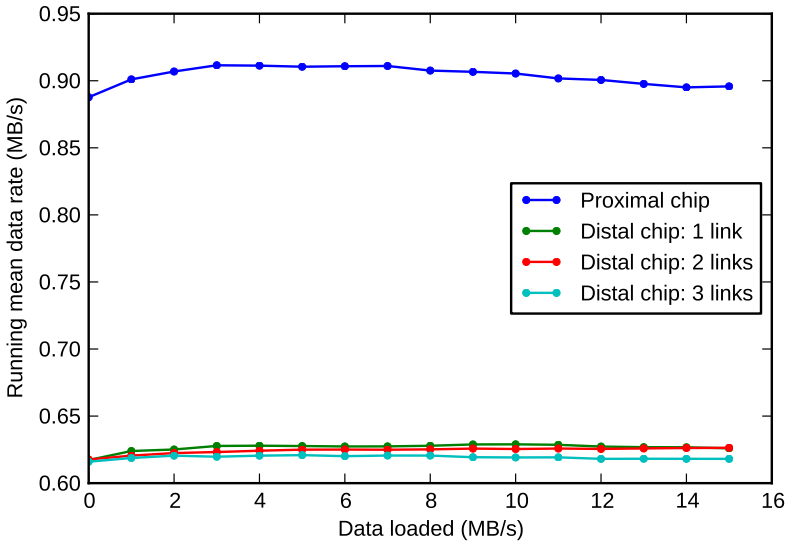


Figure 5.2: Transmission rate of SpiNNaker datagrams.

modated in instruction memory, takes approximately thirty milliseconds to cross each link and smaller programs take a linear proportion of this time. Repeated trials saw no deviation in transmission times, hence the omission of error bars. The results support the assumption, made in the previous chapter, that propagation time is invariant in the number of hops from the proximal chip, so the time required to flood-fill an entire machine may be estimated:

Navaridas *et al.* (2009) give the longest direct path between any two chips in an n-by-n triangular mesh as $\lfloor \frac{2n}{3} \rfloor$. A full-scale SpiNNaker machine, $n = 256$, should therefore boot from a single Ethernet connection in around five seconds. This figure is reduced to 150 milliseconds in the presence of an Ethernet port on each 48-chip board, which may be driven in parallel by broadcast Ethernet frames from the host.

5.1.2 Data transfer rate

The test boards were then rearranged to form the conventional triangular mesh of links between chips, in order to profile the SpiNNaker datagram protocol described in section 4.1.2. This arrangement limited the maximum distance between chips to three hops. In successive trials, arbitrary binary files of sixteen megabytes in size were transmitted by the host to chips at incremental distances from the Ethernet link as a series of 2^{14} *load memory* datagrams. The time between transmission and receipt of an acknowledgement was recorded. Figure 5.2 shows an average data rate of 0.9 megabytes per second

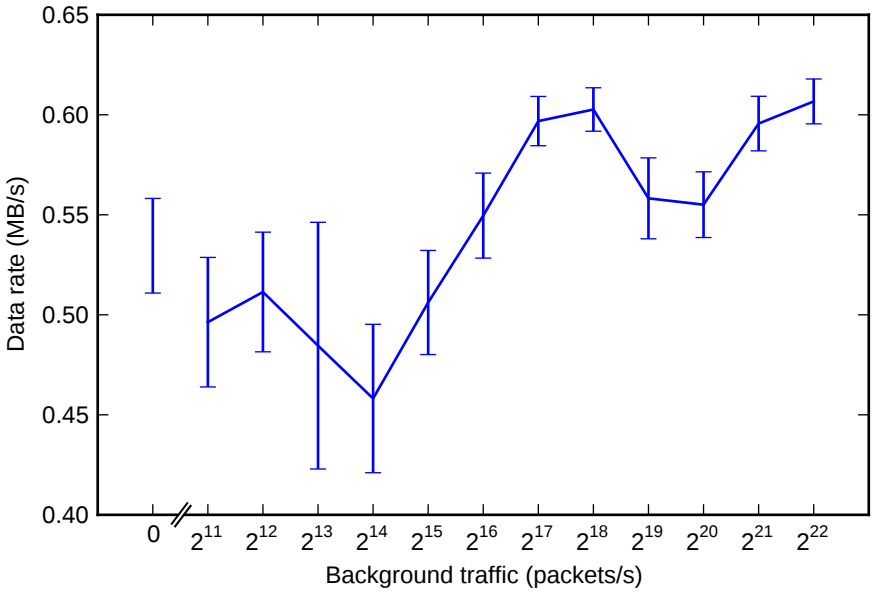


Figure 5.3: Transmission rate of SpiNNaker datagrams, with background traffic.

to the proximal chip and 0.63 megabytes per second to distal chips, largely invariant of the distance from the Ethernet connection. Again, repeated trials saw no variance.

The same arrangement of test boards was used to investigate the data rate that might be achieved during simulation. One-kilobyte datagrams were transmitted one thousand times over three interchip links, and unused processors were used to generate background traffic of up to 2^{22} packets per second in order to approximate seventeen thousand neurons per chip firing at 10Hz, as expected by Jin *et al.* (2008). The time taken to transmit and acknowledge each datagram was recorded. Figure 5.3 shows the achieved rates: the control, in which background traffic was silenced, shows a slightly reduced data rate over the previous experiment due to the proportionally increased overhead of smaller payload sizes; the tests, in which background traffic was set to between 2^{11} and 2^{22} background packets per second, show no strong correlation between data rate and router load. This suggests that an ongoing simulation with a high rate of router traffic should not adversely affect the online retrieval of intermediate results.

5.1.3 Discussion

These experiments confirm that flood-fill is fast enough to distribute homogeneous code and data to a large SpiNNaker machine. A single Ethernet port is sufficient to boot 2^{16} chips in five seconds, or a 48-chip board in 150 milliseconds.

Parameter	Unit	Value
E_L	mV	-70
V_{reset}	mV	-70
V_Θ	mV	-50
τ_m	ms	40
R_m	MΩ	50
Refrac. period	ms	1
τ_{s_e}	ms	20
τ_{s_i}	ms	5

Table 5.1: LIF model parameters for the experiments in section 5.2.

The distribution of heterogeneous data presents a greater problem. Although SDP is appropriate for transmitting small quantities, it is clearly unsuited to uploading gigabytes of synapse structures or downloading similar amounts of simulation traces. This problem is inherent in the use of massively parallel computers for simulation of neural tissue and is not unique to SpiNNaker: the volumes of data created and consumed present significant challenges to their transmission, genesis and analysis, regardless of architecture. Indeed, increasing communication bandwidth to, for example, enable speedy transfer of simulation traces from the simulator to the host only moves the problem of data analysis to a less capable computer. The solution is for the host computer to only deal in high-level representations of simulations: models to be simulated would be specified in a language such as PyNN, and this specification would be loaded onto the machine and exploded into the full neuron and synapse data structures online; analysis of simulation activity would likewise be computed online, and only statistics such as firing frequency or membrane potential correlations would be returned to the host.

5.2 Simulation accuracy

A series of experiments were performed on SpiNNaker production chips to test the function of the API and to verify the accuracy of the leaky integrate-and-fire model implementation. Experiments of increasing complexity were designed in PyNN (v0.7) and repeated on the SpiNNaker, Brian (v1.3.1) and NEST (v2.1) simulators for comparison. The LIF model was consistently instantiated with the parameters in table 5.1.

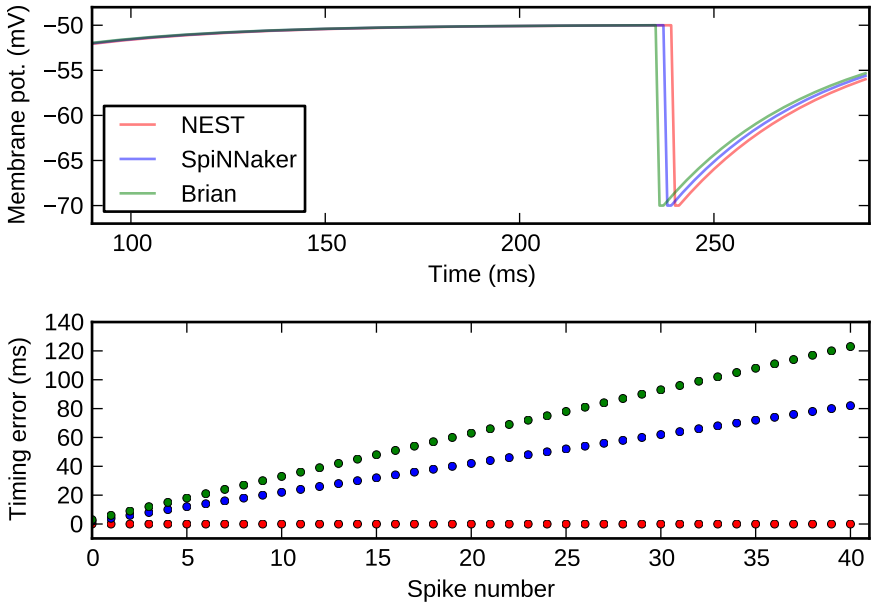


Figure 5.4: Response of an LIF neuron to rheobase current.

5.2.1 Response to rheobase current

Rheobase current is defined as the minimum constant input required to elicit a spike in a neuron. The rheobase of a biological neuron is found with a sharp electrode by applying some current for a few hundred milliseconds, checking for a spike, and repeating as necessary. The abstract nature of the LIF neuron allows for a more rigorous definition, by evaluating the analytical form as t tends to infinity

$$\lim_{t\to\infty} V(t) = E_L + R_m I - R_m I e^{-t/\tau_m}$$

$$= E_L + R_m I$$

and simply setting $V(t) = V_\Theta$ and solving for I

$$I = \frac{V_\Theta - E_L}{R_m}.$$

To elicit a spike in finite time, I was incremented by one picoamp.

A single neuron was simulated for ten seconds under rheobase current to compare error in membrane potential computations, in terms of spike times, between the three simulators and an analytical solution described by Dayan and Abbott (2001). Figure 5.4 shows the membrane potential traces from each simulator around the first spike and

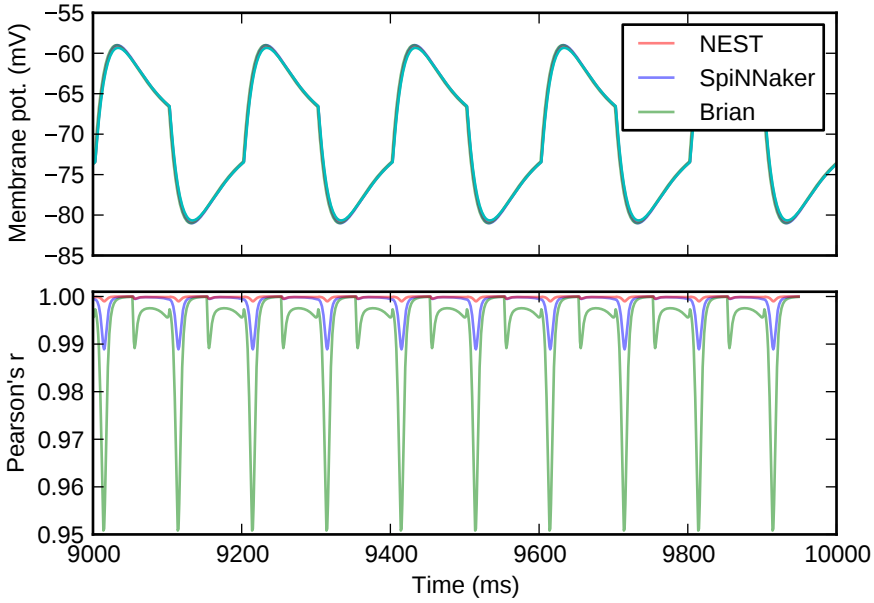


Figure 5.5: Response of an LIF neuron to regular spikes.

the cumulative error in spike times with respect to the analytical solution. It is apparent that NEST adheres to the analytically derived spike times, while a small drift in the computations performed by SpiNNaker and Brian cause firing two and three milliseconds early respectively. This drift is curtailed by the occurrence of spike, which resets the membrane potential to the common value of V_{reset}, so the timing error is discretised and successive spike times diverge linearly. In the absence of such spikes, membrane potential computations may drift more significantly; the subsequent experiment sought to test this hypothesis.

5.2.2 Response to postsynaptic currents

To evaluate computation drift over longer intervals, and to test the computation of postsynaptic currents, a single neuron was simulated under input spikes from one excitatory and one inhibitory presynaptic neuron. The interspike intervals and synaptic weights of the presynaptic neurons were tuned to ensure that postsynaptic membrane potential remained subthreshold throughout the simulation. The output of the three simulators was compared to an analytical solution presented by Brette *et al.* (2007) that, unlike the solution given by Dayan and Abbott, accounts for time-varying postsynaptic currents. The absence of output spike times for comparison required that membrane potentials were compared directly using the Pearson correlation coefficient over a sliding window

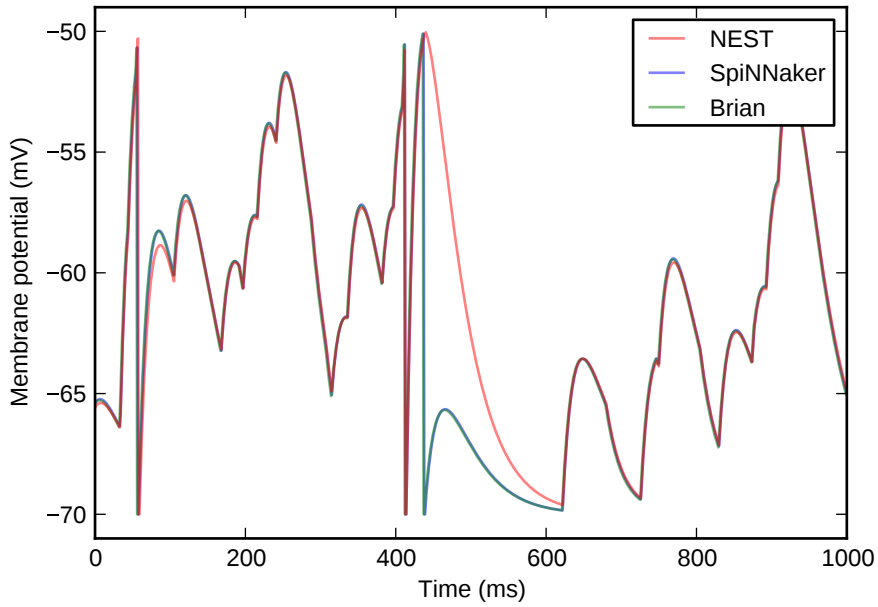


Figure 5.6: Response of an LIF neuron to random spikes.

of width equal to quarter of the input interspike interval.

Figure 5.5 shows that the membrane potential traces from all three simulators closely match the analytical solution even after ten seconds of simulation. Contrary to the hypothesis, the correlation between computed membrane potentials is strong and periodic, with a frequency equal to that of the combined input spike trains. Furthermore, the computations are robust to perturbation: figure 5.6 shows that particular patterns of input spikes may be used to elicit an erroneous output spike on certain simulators, but membrane potential traces quickly reconverge regardless.

Results from different simulators converge in this experiment because the state variables are exponentially stable and output spikes do not affect the timing of subsequent input spikes. However, in networks with recurrent projections, output spikes do contribute to subsequent inputs, which suggests that simulations of such networks on different platforms may produce greatly differing results.

5.2.3 Response to recurrent projections

To examine the effect of recurrent projections on spike times, the simple network of excitatory and inhibitory neurons shown in figure 4.8 was simulated. Excitatory neurons received rheobase current input, and projection probabilities and synaptic weights were chosen solely to cause simple oscillatory activity in the network. Although the

projections were specified probabilistically, the synapses of each projection were chosen deterministically to ensure that identical networks were deployed across all simulators.

Figure 5.7 shows the spike times and firing rates obtained from simulation on each platform. Excitatory activity is represented in blue, inhibitory activity in red, and different simulators are represented on separate axes. As expected, the results produced by each simulator differ; spikes may be easily identified that occur in one simulator and not in either of the others. Indeed, after the first five hundred milliseconds it is practically impossible to identify any spike that is common to all three simulators because the dynamics of recurrently connected networks are chaotic and, consequently, a quantitative comparison of spike times from different simulators is futile (Hanuschkin *et al.*, 2010). Qualitatively, some correlation can be observed in the firing rates of the three simulators in the first 150 milliseconds of activity, although SpiNNaker then deviates from the other two.

Networks are rarely simulated under such unstructured input. To examine firing rate correlations under more more likely input, structured stimulation was delivered to the network by an artificial spike source. A set of eighty spike trains was targeted upon the excitatory population, with a projection probability of 0.1 and synaptic weights of 0.1 nanoamps. The firing times of the spike trains was determined by a Poisson process, in which firing rate varied sinusoidally with a period of 200 milliseconds and an amplitude of 20 hertz. The PyNN interfaces to both Brian and NEST contained bugs that made such simulations impossible; for the purposes of this experiment, the PyNN-NEST interface was fixed and Brian was disregarded.

Figures 5.8 and 5.9 show key comparisons (Brette *et al.*, 2007) of the simulation results obtained from SpiNNaker and NEST. The former shows the Pearson's correlation coefficient for both the excitatory and inhibitory firing rates, calculated with a sliding window of 25 milliseconds. The latter shows the histogram of excitatory interspike intervals, taken with bin sizes of approximately five milliseconds. Both metrics show strong similarities in the outputs of the two simulators: firing rates remain strongly correlated, with only periodic divergences, after five full seconds of simulation; the interspike intervals clearly follow a similar bimodal distribution.

Brunel (2000) simulates a recurrently connected, externally driven network of leaky integrate-and-fire neurons with delta-current synapses in order to understand the fundamental properties of such models. He finds that the firing rate depends primarily on the ratio of excitatory to inhibitory synaptic currents in the network, which is a function of the number of synapses of each type and their weight. In the network discussed,

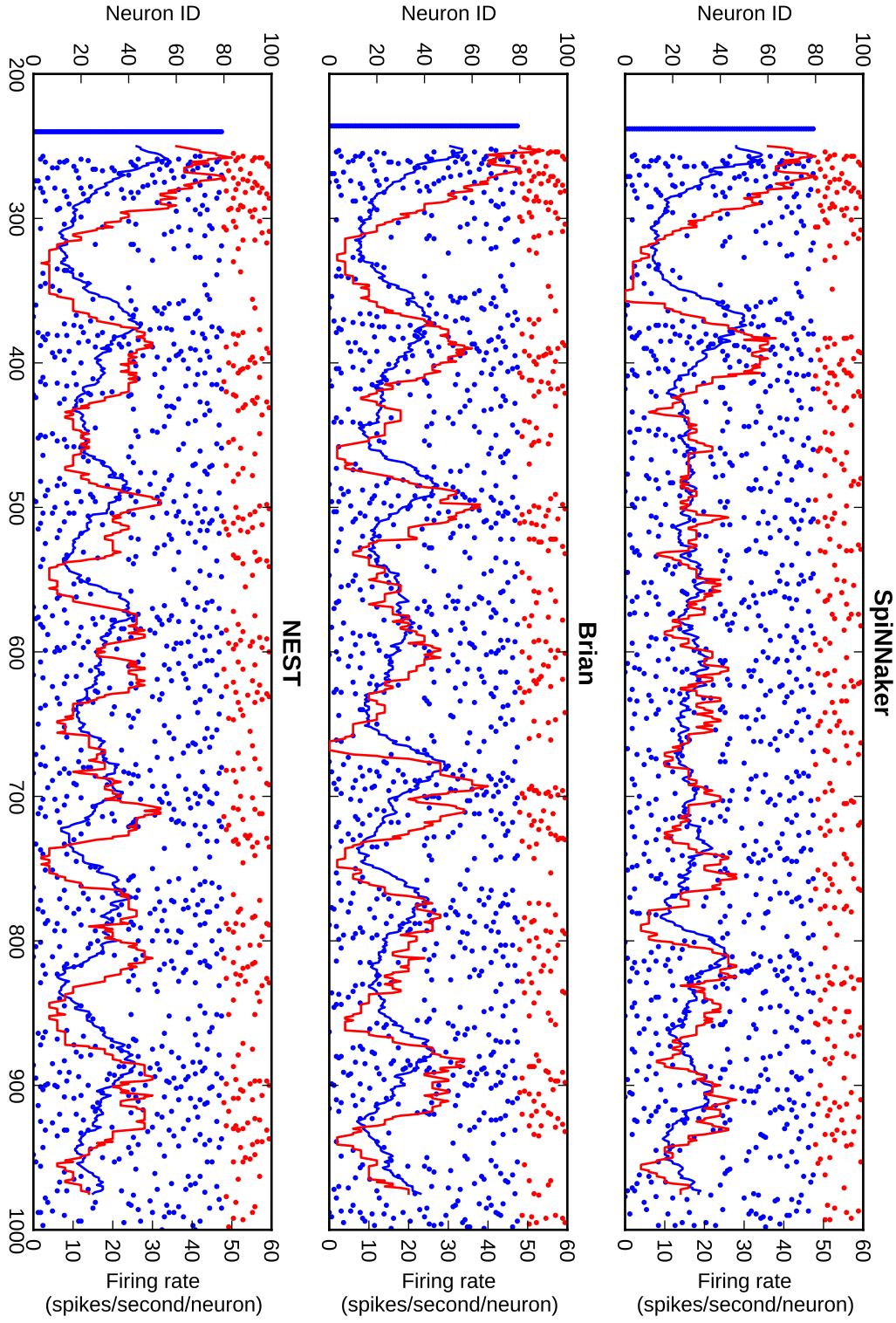


Figure 5.7: Spikes and firing rates from simulation of a simple recurrent network.

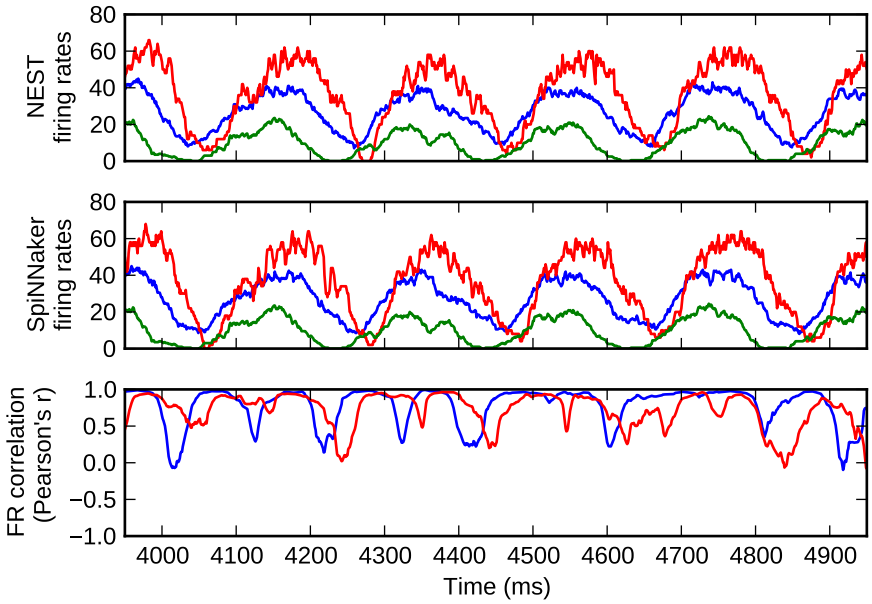


Figure 5.8: Firing-rate correlations in a simple recurrent network.

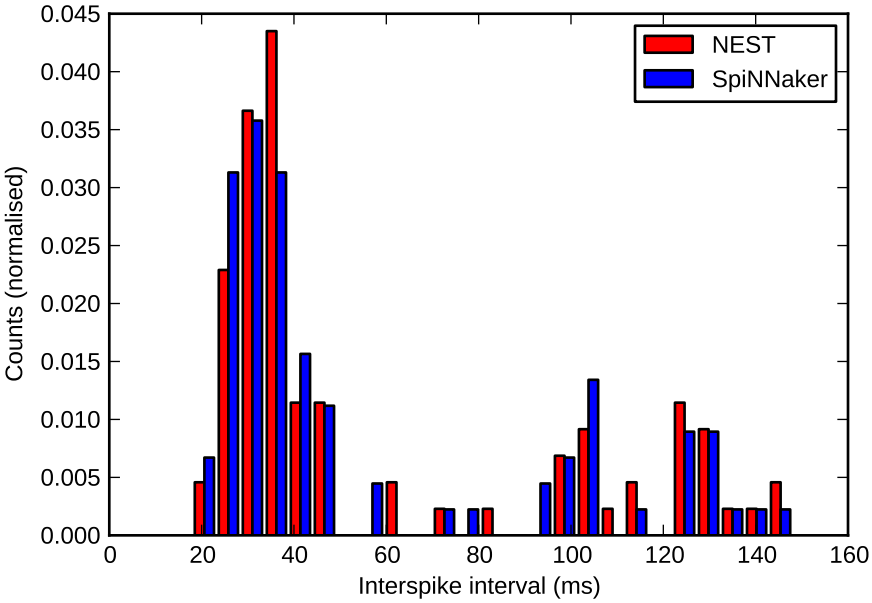


Figure 5.9: Interspike intervals in a simple recurrent network.

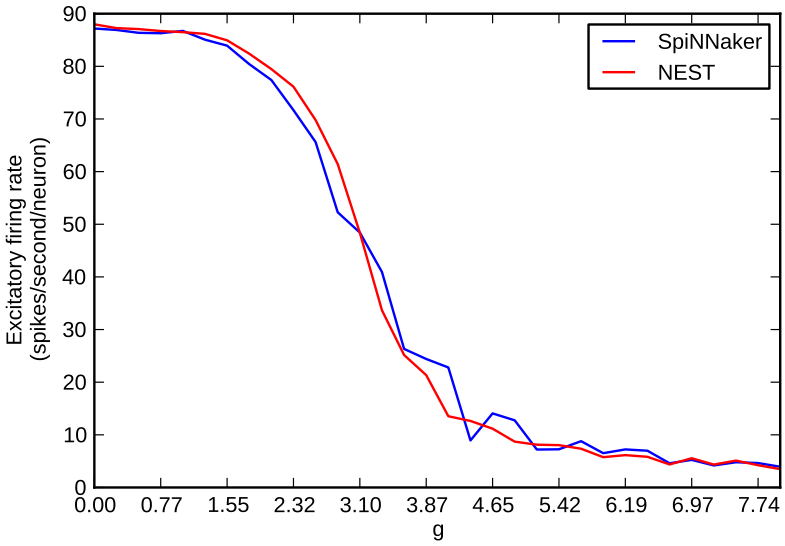


Figure 5.10: Firing rate as a function of inhibitory-to-excitatory weight ratio.

excitatory synapses outnumber inhibitory synapses four to one, so synaptic currents balance when the inhibitory-to-excitatory weight ratio g is four. When g is less than four, the network is hyperactive; when g is greater than four, the network is hypoactive.

Simulations were conducted to reproduce the findings of Brunel on NEST and SpiNNaker, using leaky integrate-and-fire neurons with exponential-current synapses. A model was constructed of eight hundred excitatory and two hundred inhibitory cells, in which every cell received a synapse from every other with a probability of 0.1 in addition to synapses from one hundred external drivers spiking at 25 hertz. Refractory periods were set to ten milliseconds to limit maximum spiking frequency. Excitatory weights were fixed at 0.1 nanoamps and inhibitory weights were varied with a value of g ranging from zero to eight in successive trials. The average firing rate across all excitatory cells was recorded.

Figure 5.10 shows the firing rates achieved by SpiNNaker and NEST, in their usual colours, as a function of the weight-ratio coefficient g. The simulators produce results in agreement with both Brunel and one another, in that firing rate varies sigmoidally as a function of g from near-maximum to near-minimum.

5.2.4 Discussion

These experiments show that SpiNNaker simulations are faithful to results obtained from established simulators. Under constant current input current, SpiNNaker evalu-

ates neuron equations almost as accurately as NEST and more so than Brian. The same may be said for periodic exponential synaptic currents, where there is is no long-term divergence in results from each simulator. Under asynchronous nonrecurrent synaptic inputs, SpiNNaker may emit a spike and reset membrane potential where NEST does not, but the exponentially stable neuron equations soon reconverge. Recurrent synaptic connections, however, amplify any differences in membrane potential evaluation, so spike-by-spike comparison of results across simulators becomes impossible (Hanuschkin *et al.*, 2010). This also manifests in higher-order observations of activity, in that there is little correlation between the firing rates of recurrently connected networks under unstructured input simulated on SpiNNaker, NEST and Brian. However, under structured input, networks simulated on SpiNNaker and NEST show strongly correlated firing rates and qualitatively similar distributions of interspike intervals. In parameter-sweeping experiments, these simulators agree with both one another and the literature that network firing rate is a function of the ratio between inhibitory and excitatory synaptic-current amplitudes. So, taking NEST as a reference point, SpiNNaker appears to be a reliable and correct simulator of recurrently connected networks.

5.3 Performance profile

A series of experiments were conducted to sketch a performance profile of SpiNNaker hardware and software (Sharp and Furber, 2013). Each SpiNNaker processor has a *core thread* for executing software and a *DMA thread* for transferring data between memories. Simulations of neural tissue comprise neuron state computations that are performed exclusively by the core thread and synapse state computations that are performed by a pipeline through the core and DMA threads. As shown in figure 4.4 the three-stage packet-processing pipeline begins with receipt by the core of a spike packet from the router, proceeds with a DMA transfer of corresponding synaptic weights to processor-local data memory, and concludes with translation by the core of synaptic weights into input currents to neurons. It is apparent *a priori* that this pipeline dominates core time: computing each neuron state costs approximately 2^7 instructions, responding to each *synaptic event*, defined as one spike innervating one synapse, requires 2^5 instructions; each processor must handle 2^7 neuron state computations and 2^{13} synaptic events per millisecond, so the latter clearly dominate. The most important performance profile for SpiNNaker, therefore, is that of the packet-processing pipeline.

A simple PyNN network was constructed in which a source population on one

processor projected to target populations on one or more processors of the same chip. In each trial, the source population was configured to emit some number of spikes in every millisecond, which innervated some number of synapses on each of the target populations. At the start of each millisecond, each target processor counted and flushed the *excess* packets that were left unprocessed from the previous millisecond; these data were reported to the host at the end of the simulation. From the parameters and excess counts of each trial, it was possible to infer the pipeline throughput.

Jin *et al.* (2008) set a per-processor performance target of 1,024 optimised Izhikevich neurons with delta-current synapses and ten thousand synaptic events per millisecond. The leaky integrate-and-fire neurons with exponential-current synapses used here are fourfold as costly, so only 256 cells are simulated per processor. Successive trials explored a parameter range of 0 to 128 presynaptic spikes and 0 to 512 postsynaptic synapses, in steps of 16 and 32 respectively. The former parameter controlled the number of DMA transfers to be made and processed, and the latter controlled the size of each transfer and the consequent cost of processing. In order to examine each stage of the pipeline, the simulated network was contrived to offer three modes of processing each spike: immediately discarding the spike; performing a DMA transfer of synaptic weights to data memory then discarding the transfer results; and performing a DMA transfer of synaptic weights to data memory then processing the transfer results into postsynaptic currents, as in a conventional simulation. Checks were performed to ensure that no packets were lost in transit between processors and that callback queues did not overflow. Callback priorities were chosen as simply as possible, such that the packet callback was priority minus one (preemptive) and all others were priority one (queueable); this ensured that packets were cleared from the communications controller immediately upon receipt and other callbacks were executed in the order in which their causative events occurred. The DMA callback was set to process the weights of the causative DMA transfer before setting up the next transfer; the term *pipeline* should not be interpreted, as in the field of computer processor design, to mean that each stage was utilised in parallel. The processors were run at 200 megahertz and the router, system bus and off-chip memory were run at 133 megahertz.

5.3.1 Packet processing cost

The first stage of the packet-processing pipeline accepts a spike packet from the communications controller and looks up the address of the corresponding synaptic weights in off-chip memory; if none are found, the packet is simply discarded.

A projection was contrived between the source and target populations in which all synapse lookups failed immediately, thereby testing the first stage of the processing pipeline alone. Under these conditions, no excess packets were observed in any trial.

5.3.2 Direct memory access cost

In the second stage of the packet-processing pipeline, a DMA transfer is committed on the basis of the lookup results and the DMA engine copies weights while the processor is free to perform other computations.

A projection was created in order to commit a DMA transfer on receipt of a spike then dispose of the copy results on completion, thereby testing the first two stages of the processing pipeline in isolation. Figure 5.11 shows the worst observed excesses from all processors in each trial, superimposed with the constant product $xy = 10^4$ that represents the architectural target of ten thousand synaptic events per millisecond. DMA bandwidth is clearly sufficient for a single processor to work far in excess of the architectural target, and is almost sufficient to service fifteen processors in parallel.

Considering the latter case and the data point of twenty spikes innervating five hundred synapses, each of the fifteen processors retrieves ten thousand four-byte synapses per millisecond, which makes for a total throughput of six hundred megabytes per second. This observation of maximum DMA throughput is in direct agreement with results obtained by Painkras *et al.* (2012). Note that excesses are considerably greater at the other end of the target curve: although the number of synaptic events under consideration is the same, the greater number of input spikes and DMA transfers makes for a significant increase in overhead.

Ultimately, it is clear that DMA bandwidth is a significant determinant of pipeline throughput. According to Painkras *et al.* each processor may retrieve around two hundred megabytes per second via DMA, so three processors operating at their maximum transfer rates will saturate the data bus. If four or more processors access the bus concurrently, they must share the available bandwidth and will therefore see a diminished throughput. Thus, the throughput observed in the right panel of figure 5.11 is the maximum achievable by the packet-processing pipeline.

5.3.3 Postsynaptic computation cost

In the final stage of the packet-processing pipeline, the copied weights are read iteratively and processed into postsynaptic currents for their target neurons.

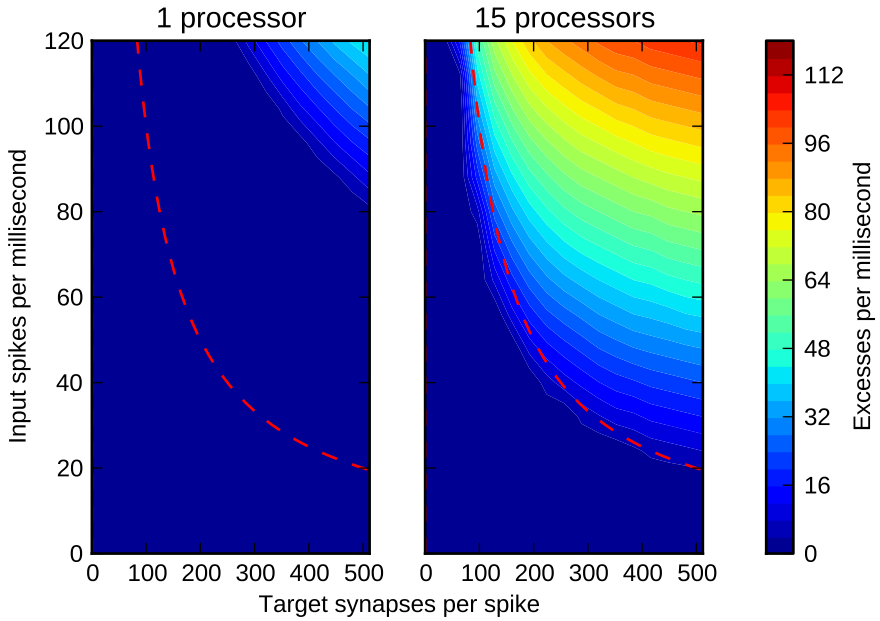


Figure 5.11: Packet excesses without postsynaptic currents.

A projection was created in order to commit a DMA transfer on receipt of a spike and then process the copied synaptic weights as normal, thereby testing the entire pipeline. Figure 5.12 again shows the worst packet excesses in comparison to the architectural target. Performance in both the single- and fifteen-processor cases drops to well below the curve, to almost exactly the same value. This suggests that postsynaptic computation so dominates the packet-processing pipeline that the cost of DMA transfers is negated.

A coarse estimate of processor utilisation during the millisecond simulation cycle is telling of the results obtained here. A processor running at 200 megahertz may execute 200,000 single-cycle instructions in a millisecond. According to the disassembled simulation code, each leaky integrate-and-fire neuron requires approximately 128 of these processor cycles and each synaptic event needs around 32. Simulation of 256 neurons consumes 30,000 cycles, which leaves approximately 170,000 cycles for handling synaptic events. As such, it should be possible to process around 5,000 synaptic events per millisecond; this curve is plotted in green on figure 5.12 and agrees with the observed excesses remarkably well, barring the overhead-asymmetry discussed previously.

The packet-processing pipeline throughput appears therefore to be defined entirely in terms of core-thread performance. This explains why performance does not differ between the single- and fifteen-processor cases, but it raises the question as to why

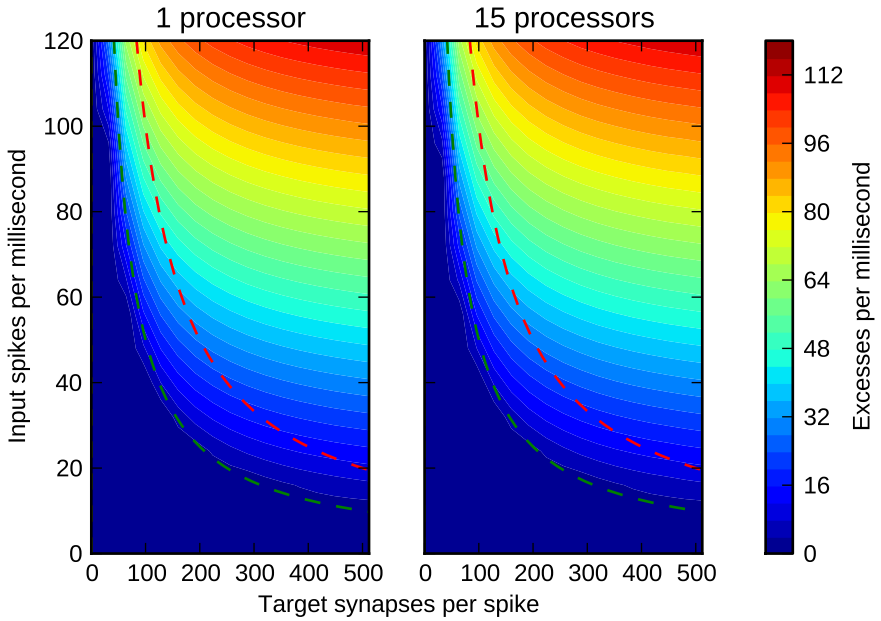


Figure 5.12: Packet excesses with postsynaptic currents.

DMA-transfer time does not factor in the results. The DMA-transfer and postsynaptic-processing stages occur consecutively in the packet-processing pipeline, so the time to complete the former step must be vanishingly small if it is not to manifest in the performance data. Indeed, a transfer of n four-byte synapses at two hundred megabytes per second will take

$$\frac{4n\text{B}}{2 \cdot 10^8\text{B/s}} = 2n \cdot 10^{-8}\text{s}$$

while processing the same quantity will take

$$\frac{32n\text{C}}{2 \cdot 10^8\text{C/s}} = 16n \cdot 10^{-8}\text{s}$$

where C denotes processor cycles. As such, the DMA stage of the packet-processing pipeline takes only around a tenth of the total time, which is well within the error margins of the coarse utilisation estimates above. This explains why the performance of one processor, achieving the maximum DMA throughput, may be defined almost entirely in terms of processor utilisation. To explain the similar performance of the fifteen-processor case, it must be considered that only one tenth of each processor's time is spent making DMA transfers; on average only two of the eighteen processors are making transfers at any time, so both see their full two-hundred-megabytes-per-second throughput without saturating the data bus. This argument boldly assumes that

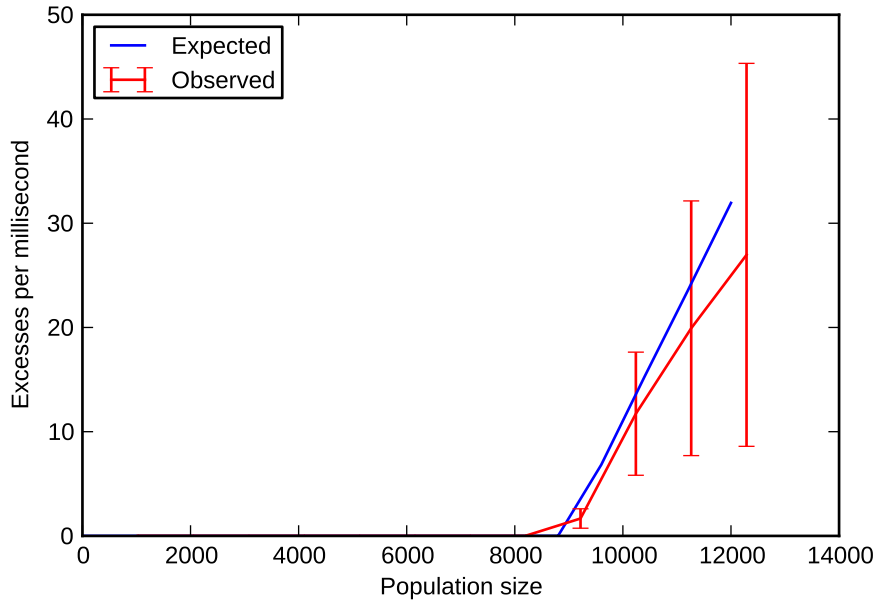


Figure 5.13: Actual and expected excesses in a controlled network simulation.

DMA transfers are uniformly distributed throughout the millisecond; this cannot be proven from the data but it is a likely interpretation of the results.

5.3.4 Validation

A final experiment was conducted to validate the results above. A single population of trial-varying size was configured such that each neuron fired at ten hertz, and initial membrane potentials were distributed so that firing was uniformly spread through time. The N neurons in each trial were split across $\lceil \frac{N}{256} \rceil$ processors. A recurrent projection was made from the population onto itself with probability $p = 0.1$ and synaptic weight 0, in order that received spikes would induce postsynaptic-current computations but have no affect on the established regular spiking. Each processor therefore computed $256pN \cdot 10^{-3}$ synaptic events per millisecond.

Figure 5.13 shows the mean and variance in excesses across all processors as a function of population size N. The blue line shows the expected number of packet excesses according to the results in figure 5.12. Although the standard deviations are large, the means of the observed results are in agreement with those expected, which suggests that the earlier experiments accurately model the packet-processing pipeline in simulations of recurrent networks.

5.3.5 Execution time

A complete SpiNNaker system consists of the core and DMA threads of each processor, which have been profiled above, and a *host thread* that constructs and controls simulations. As such, the total execution time of a simulation is a function of both host and chip performance. SpiNNaker simulations typically run in real-time, but the host machine may spend additional time generating and uploading data structures and downloading and processing results. The performance profile of the system as a whole, and its relation to comparable simulators, is therefore of significant interest.

The Vogels-Abbott benchmark is a network of excitatory and inhibitory neurons, much like that simulated in section 5.2, that is conveniently encoded in a PyNN script by Brette *et al.* (2007). The benchmark was run on SpiNNaker and NEST (v2.1) with parameters ranging from 2^9 to 2^{13} neurons and 2^2 to 2^6 simulated seconds. The SpiNNaker benchmarks used one host processor to control simulations and $\lceil \frac{N}{256} \rceil$ SpiNNaker processors to execute them; the NEST benchmarks were run firstly on one core of the host machine, and then repeated on a high-performance multicore machine. The host machine was based upon a triple-core AMD Athlon-II X3-445 processor clocked at three gigahertz served by four gigabytes of memory; the multicore machine was based upon a machine with 130 gigabytes of memory and 4 12-core AMD Opteron 6174 processors clocked at 2.2 gigahertz; the SpiNNaker board contained four chips, totalling 72 processors, with processor clocks set to 200 megahertz and router, system bus and off-chip memory clocks set to 133 megahertz. The Python `cProfile` module was used to record the execution profile of each simulation. The underlying simulation engine of NEST is written in C++ and is significantly faster than the Python-based Brian simulator; as such, Brian was not included in the performance comparisons.

Figure 5.14 shows execution time of a 32-second simulation on NEST using the host machine and on SpiNNaker, as a function of neuron count. SpiNNaker beats NEST in all cases beyond 1,024 neurons. Run time on SpiNNaker remains a constant 32 seconds, and time to prepare data structures, load them and dump simulation results grows with model size. Figure 5.15 shows execution time of a 4,096-neuron model as a function of simulation time. Here, SpiNNaker beats NEST in all cases. Run time on SpiNNaker grows as the identity of simulation time, preparation and loading time remains constant, and dump time grows with simulation time.

The performance profile of SpiNNaker is easily explained. Simulations execute in real-time, so run time is consistently equal to the simulation period. Preparation and load times are a quadratic function of model size: synapse structures comprise the ma-

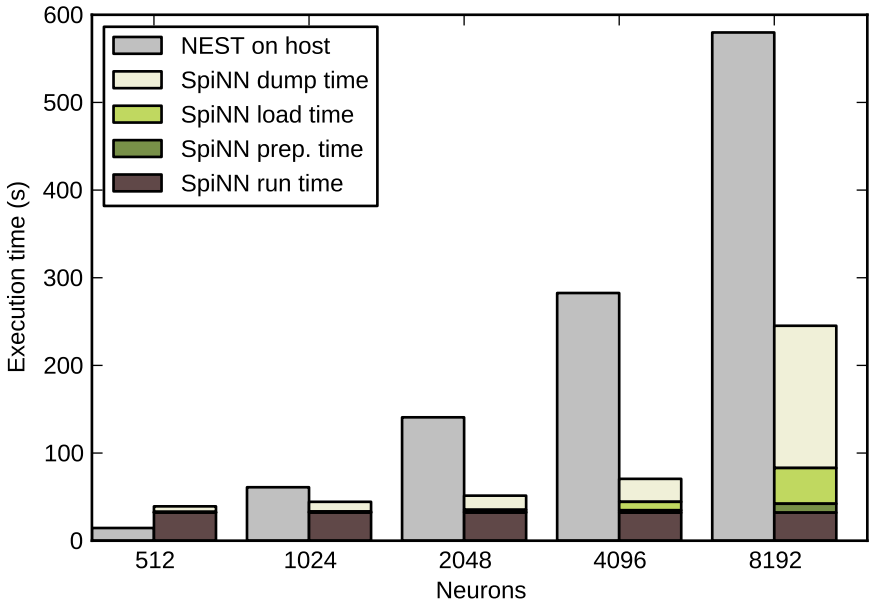


Figure 5.14: Execution time of the 32-second Vogels-Abbott benchmark.

jority of data to be generated and sent to the machine, and a recurrently connected network of size N with projection probability p forms pN^2 synapses. Dump time comprises the cost of retrieving and sorting spikes from the simulation, and is some function of model size, firing rate and run time. However, both the bandwidth and software exist for spikes to be dumped via Ethernet in parallel with the simulation, as shown in the prior and subsequent experiments respectively, so this cost may be entirely disregarded.

Performance comparisons between simulators are often complicated by vast differences in peak hardware throughput; as argued in section 3.2.4 achieving 2.5× speedup from an architecture that is 3× more powerful than its counterpart is rather unremarkable. The sixteen application processors of a SpiNNaker chip have a combined total throughput of $3.2{\cdot}10^9$ operations per second, which is similar to the $3{\cdot}10^9$ operations per second available to one core of the host processor. Naturally, there are innumerable differences in the hardware and software architectures of the two platforms, some favourable to one and some to the other, but neither excels purely in terms of arithmetic throughput. The 4,096-neuron simulations presented in figure 5.15, which use sixteen processors, are therefore a reasonably fair comparison of SpiNNaker and NEST. Disregarding dump time, as justified above, the former achieves a sixfold speedup over the latter at all points beyond 4-second simulations.

Figure 5.16 reports the run time of NEST for 32-second, 8,192-neuron simulation

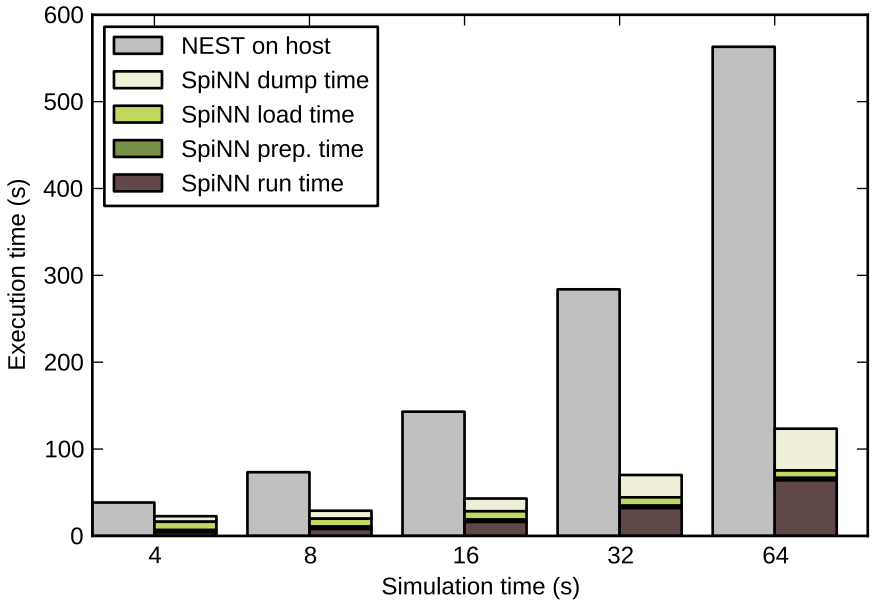


Figure 5.15: Execution time of the 4,096-neuron Vogels-Abbott benchmark.

as a function of the number of threads used on the high-performance machine, alongside the execution profile of the same simulation on two SpiNNaker chips. On a single thread, NEST is five times slower than SpiNNaker, disregarding dump time. The application of two threads to NEST simulation provides an almost $2\times$ speedup but further speedup is sublinear in terms of the number of threads, such that NEST is incapable of bettering SpiNNaker with any number of processors. Furthermore, NEST draws upon approximately three times the computational resources and fifty times the power of SpiNNaker: two SpiNNaker chips draw two watts to provide six gigaops per second, whereas the 12 processors of an Opteron chip draw 115 watts to provide approximately 20 gigaops per second (AMD, 2013).

5.3.6 Discussion

These experiments build a detailed performance profile of SpiNNaker simulations at both the processor and system levels. Results show that each processor may simulate 256 neurons with complex synaptic currents in real-time while processing 5,000 synaptic events per millisecond. A 32-second simulation of an 8,192-neuron network may be completed in less than 100 seconds, six times faster than NEST running on a single processor. A 64-second simulation of 4,096 neurons completes in a similar time with a similar speedup. The performance of NEST improves with additional threads

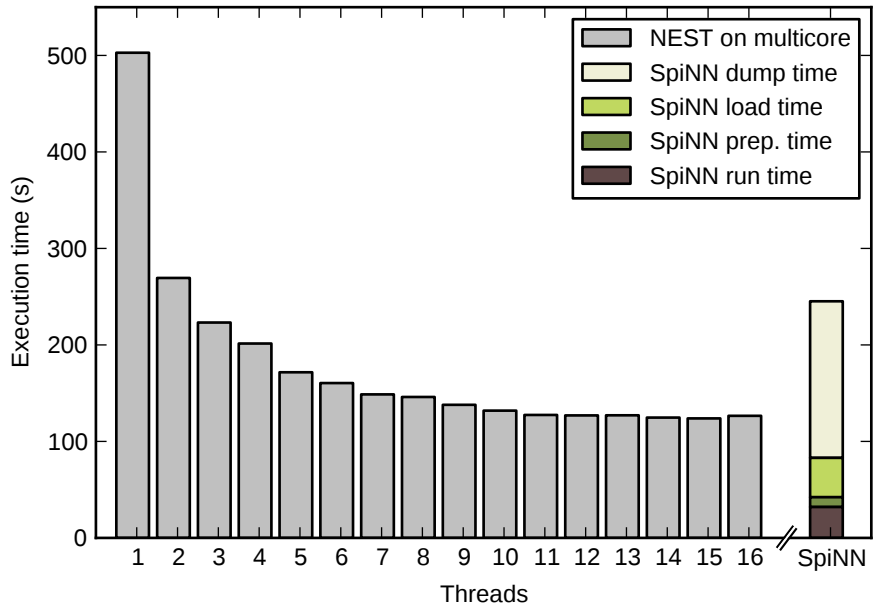


Figure 5.16: Execution time of the 32-second Vogels-Abbott benchmark.

but suffers sublinear scaling, such that it is unable to better SpiNNaker even with the application of five times the computational resources and fifty times the power.

Examination of the packet-processing pipeline shows that a processor can handle 5,000 synaptic events per millisecond. Stromatias *et al.* (2013) somewhat support these results with network simulations, not tuned for peak throughput, that span a full board of 768 processors to generate at least 1.5 gigaevents per second, thereby averaging two megaevents per second per processor. The results are also comparable to those of Fidjeland and Shanahan (2010) who simulate networks of increasing size in order to understand the achievable synaptic-event throughput of the NeMo simulator on an Nvidia Tesla C1060 graphics processing unit (GPU). Figure 5.17 reproduces the best results in figure 3 of the referred work and plots SpiNNaker performance for comparison according to the equation

$$y = 5 \cdot 10^6 \cdot \frac{x}{256}$$

where the fraction is the number of processors used to simulate x neurons, assuming 256 neurons per processor, and the coefficient is the synaptic-event throughput per second per processor; the equation assumes linear throughput scaling with processor count, since the experiments above show that processors enjoy contentionless access to memory and because interprocessor communication bandwidth is effectively inexhaustible (Navaridas *et al.*, 2009). NeMo simulates small networks faster than real-

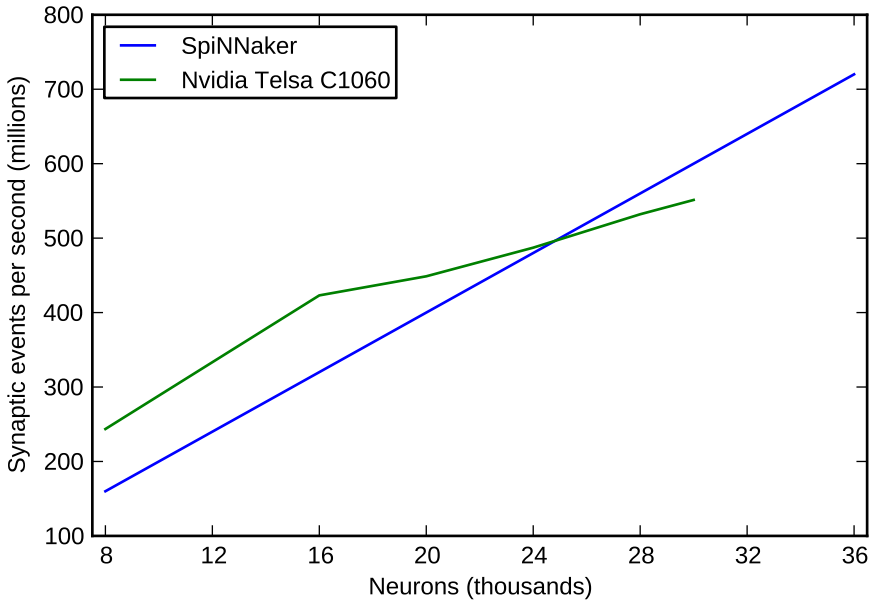


Figure 5.17: SpiNNaker performance versus GPU (Fidjeland and Shanahan, 2010).

time (see reference for simulation methodology) and thereby outperforms SpiNNaker on these scales, but the platform suffers sublinear throughput scaling with increasing network size and, regardless, is limited to thirty thousand neurons on one GPU. SpiNNaker overtakes the GPU-based simulator at around 24k neurons, which is equivalent to around one hundred SpiNNaker processors. However, one hundred SpiNNaker processors clocked at two hundred megahertz offer twenty gigaops per second of computational throughput on a budget of less than ten watts, and are therefore significantly more efficient than the C1060 GPU that provides approximately one teraflop per second at the expense of almost two hundred watts (Nvidia, 2013).

The performance data in figures 5.11, 5.12 and 5.13 accurately describe the maximum processor throughput and clearly identify the pipeline bottleneck to be optimised. Unfortunately, the loop that dominates processor time is only 32 instructions long, so there is little room for improvement; careful programming may save three or four instructions and improve performance by one tenth to one eighth. The ARK may be optimised to reduce overhead in the many-spikes-few-synapses domain, but this involves difficult programming and further profiling; effort would be better spent on exploiting the abundant parallelism of the production SpiNNaker boards, and minute optimisations should not be considered until every available processor is fully utilised.

The API allows callbacks to be registered against the ARK with arbitrary priorities.

Tangential experiments to those reported upon here show that the priorities used in the performance profiles are in-fact necessary for correct simulation of neural tissue. The packet-received callback must be preemptive to ensure that processors immediately clear packets from the communications controller, which will otherwise clog the routers. The DMA callback must not be set to greater priority than the timer callback, lest a regular excess of packets cause the former to dominate processor time and indefinitely postpone the latter; both should be set to equal, queueable priority.

The execution profile of the Vogels-Abbott benchmark shows that SpiNNaker outperforms NEST in all nontrivial simulations. Simulation of 4,096 neurons on SpiNNaker draws upon similar computational resources as the same simulation on NEST using one host processor, yet the former shows a sixfold speedup over the latter. NEST performance improves with additional threads, but suffers sublinear scaling such that it is unable to better SpiNNaker even with overwhelming computational resources. The SpiNNaker results demonstrate that load time becomes significant as models scale up, as predicted by earlier experiments, so high-bandwidth channels and data compression will be required to fuel larger simulations. Dump time increases likewise, but subsequent simulations show that results may be extracted from the machine in real-time, which comprehensively solves the problem. The NEST simulation kernel is written in C++, which the `cProfile` is unable to report upon in detail, so further analysis of performance is not possible. However, the data is sufficient to show that SpiNNaker significantly accelerates simulation of models on the order of thousands of neurons. Furthermore, the data suggest that the performance gap between the two simulators will continue to grow linearly as models scale into tens of thousands of neurons.

5.4 Power profile

A series of experiments were performed to draw a power profile of simulations on SpiNNaker (Sharp *et al.*, 2012). The availability and cost of electrical power is a significant constraint on the volume of neural tissue that may be modelled, so it is important to understand the power requirements of any large-scale simulation platform. Each SpiNNaker chip should consume around one watt (Furber and Brown, 2009) so a full scale machine should consume less than 10^5 watts. To investigate this prediction, a common model of the cortex was simulated across four SpiNNaker chips and detailed observations of power consumption were made. The processors were run at 150 megahertz and the the router, system bus and off-chip memory were run at 100 megahertz.

Population	Neurons	Synapses
nb1	150	24,150
p2/3	2,600	1,666,599
b2/3	310	103,850
nb2/3	420	116,760
ss4(L4)	920	401,120
ss4(L2/3)	920	331,200
p4	920	430,560
b4	540	114,479
nb4	150	37,500
p5(L2/3)	480	221,760
p5(L5/6)	130	98,540
b5	60	15,780
nb5	80	21,040
p6(L4)	1,360	522,240
p6(L5/6)	450	125,100
b6	200	22,000
nb6	200	22,000
Total	9,890	4,284,568

Table 5.2: Neurons and synapse counts in visual-cortex model populations.

5.4.1 Model of the visual cortex

A model of the cat visual cortex was constructed according to composition and connectivity data by Binzegger *et al.* (2004), as reproduced by Izhikevich and Edelman (2008, supporting material). These data were chosen for their complexity and prevalence in the simulation literature, as discussed in chapter 3, such that the model should be comparable to existing studies and sufficiently intricate to stress SpiNNaker.

The model was constructed of ten thousand neurons and four million synapses, distributed across seventeen populations. The proportional sizes of the populations were governed by the data, as were the proportions of synapses received by each population from each population. The absolute numbers of synapses on each neuron were not preserved as they would form absurdly dense connectivity in a circuit of just ten thousand neurons. Consequently, the synapses counts listed by Izhikevich and Edelman were translated into projection probabilities p_{ij} from population j to population i for use in

PyNN according to

$$p_{ij} = c \cdot \frac{S_i s_{ij}}{N_j}$$

where N_j is the number of neurons in the j^{th} population, S_i is the total number of synapses received by a neuron in the i^{th} population, s_{ij} is the proportion of those synapses received from neurons in the j^{th} population, and $c = 0.1$ is an arbitrary sparsification coefficient. Where data were listed for multiple dendritic compartments of the same cell, the values were summed to represent a single compartment. Projections with a probability of less than 0.01 were pruned. Table 5.2 lists the total numbers of neurons and synapses in the model and table 5.3 contains the matrix of projection probabilities between populations. Layer identifiers in brackets indicate the primary target of innervation for that population thereby distinguishing, for example, layer 5 pyramids that project into layers 2 and 3 from those that project into layers 5 and 6.

Neurons were simulated using the Izhikevich (2003) model

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I$$

$$\frac{du}{dt} = a(bv - u)$$

$$\text{if } v \geq 30\text{mV}: \; v \leftarrow c, \; u \leftarrow u + d$$

where a, b, c and d were chosen to determine appropriate spiking dynamics for the neuron types in each population: excitatory pyramidal cells (denoted pl for pyramids in layer l) and stellate cells (ss$l$) were simulated as regular spiking neurons, and inhibitory basket cells (bl) and nonbasket (nb$l$) cells were simulated as fast spiking neurons.

The postsynaptic spike-response was simulated as an exponential current, with amplitude w and time constant τ for each type of synapse. Synapse parameters are listed in table 5.4. As with synaptic sparseness, synaptic weights were chosen *ad hoc* to ensure stable network activity; time constants, however, were in accordance with those suggested by Thomson *et al.* (2002).

To induce spiking activity in the network, background synaptic currents were simulated using a gross approximation of an Ornstein-Ohlenbeck process inspired by Destexhe *et al.* (2001). Each neuron received a time-varying background current $I_b(t)$ which was computed as

$$I_b\big(t+1\big) = \bar{I_b^C} + \tau\big(I_b(t) - \bar{I_b^C}\big) + A \cdot U\big(-0.5, +0.5\big)$$

		Presynaptic population																
		nb1	p2/3	b2/3	nb2/3	ss4(L4)	ss4(L2/3)	p4	b4	nb4	p5(L2/3)	p5(L5/6)	b5	nb5	p6(L4)	p6(L5/6)	b6	nb6
Postsynaptic population	nb1	.600	.022	.016	.024	.	.	.	.	.	.	.	.	.	.	.	.	.
	p2/3	.087	.137	.171	.064	.	.043	.049	.	.033	.090	.	.	.	.010	.	.	.230
	b2/3	.033	.077	.132	.031	.	.024	.027	.	.020	.050	.	.	.	.	.	.	.015
	nb2/3	.040	.062	.123	.026	.	.020	.023	.	.020	.042	.	.	.	.	.	.	.
	ss4(L4)	.	.	.	.	.075	.023	.026	.076	.080	.096	.	.	.	.139	.	.	.170
	ss4(L2/3)	.	.011	.	.	.061	.021	.024	.067	.073	.013	.	.	.	.114	.	.	.135
	p4	.053	.031	.016	.017	.063	.026	.030	.067	.073	.027	.	.	.	.118	.	.	.185
	b4	.	.	.	.	.039	.013	.015	.050	.053	.	.	.	.	.072	.	.	.085
	nb4	.	.	.	.	.047	.014	.016	.056	.060	.	.	.	.	.087	.	.	.105
	p5(L2/3)	.013	.087	.032	.	.020	.014	.040	.	.033	.113	.031	.067	.062	.015	.020	.	.275
	p5(L5/6)	.387	.135	.052	.031	.032	.026	.057	.	.047	.144	.054	.067	.062	.035	.029	.	.320
	b5	.	.052	.023	.	.011	.	.024	.	.020	.073	.023	.050	.050	.	.013	.	.170
	nb5	.	.052	.023	.	.011	.	.024	.	.020	.073	.023	.050	.050	.	.013	.	.170
	p6(L4)	.	.027	.032	.	.032	.014	.024	.019	.027	.031	.131	.017	.025	.052	.100	.125	.245
	p6(L5/6)	.	.017	.	.	.	.	.014	.	.020	.013	.215	.	.025	.012	.164	.215	.240
	b6	.	.	.	.	.	.	.	.	.	.	.123	.	.013	.	.096	.125	.125
	nb6	.	.	.	.	.	.	.	.	.	.	.123	.	.013	.	.096	.125	.125

Table 5.3: Projection probabilities between visual-cortex populations.

		Pre	
		Exci.	Inhi.
Post	Exci.	$w = 0.05$, $\tau = 30$	$w = 0.05$, $\tau = 30$
	Inhi.	$w = 0.20$, $\tau = 15$	$w = 0.10$, $\tau = 15$

Table 5.4: Synapse parameters of the visual cortex model.

where $\bar{I_b^C}$ is the mean background current for cell class C, τ is proportional to the process time constant, and A is the amplitude of the fluctuations drawn from the uniform random distribution U. Mean background currents were set to $\bar{I_b^p} = 2.8$ for pyramids, $\bar{I_b^b} = 2.8$ for baskets, $\bar{I_b^{nb}} = 0.5$ for nonbaskets. The process time constant and amplitude fluctuations were set to $\tau = 0.9$ and $A = 0.85$. Pseudorandom numbers were drawn from U using a software linear feedback shift register (ARM, 1995). Again, $\bar{I_b^C}$ was chosen for stable network activity, not to reflect known values of biological neurons. Thus, $\bar{I_b^C}$ efficiently approximates the stimulus method employed by Izhikevich and Edelman (2008) and Phoka *et al.* (2012) in which network activity is induced by random, spontaneous excitatory potentials on every synapse.

5.4.2 Experimental protocol

The cortical microcircuit was simulated on a four-chip test board. PACMAN allocated neural populations to fifty processors and assigned a further four, one on each chip, to aggregate spikes from neighbouring processors and forward them via the monitor processors to the host in real-time; the remaining application processors were left idle. The average processor load of two hundred neurons was intentionally made light to exploit the abundant available parallelism, rather than labouring to squeeze all possible performance from fewer cores.

An oscilloscope was used to record the power consumed during simulation. Resistors of 0.1 Ohms were placed on the 1.2 volt and 1.8 volt power rails, which supply the SpiNNaker chips and the off-chip memories respectively. The voltage drop across these resistors was measured with the oscilloscope and the dynamic power consumption was computed. The oscilloscope could only store 45,000 samples per trial, so the simulation was observed for 45 milliseconds with a one microsecond sampling period.

A series of experiments were conducted to determine the particular power costs of simulation in an arbitrarily chosen period from $t = 5{,}100$ milliseconds to $t = 5{,}145$ milliseconds. In the first experiment, the simulation was run as normal and the power

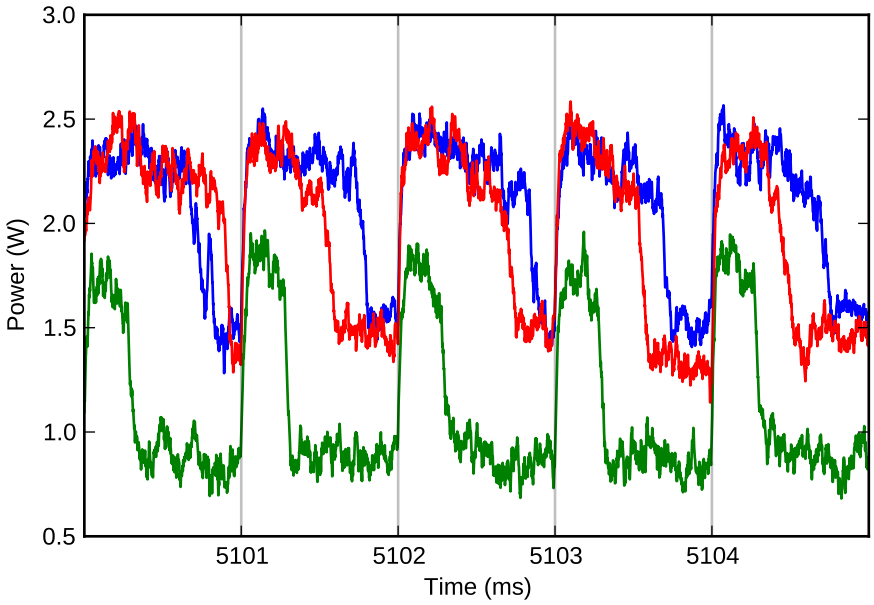


Figure 5.18: Power traces from full and simplified simulations.

consumption was recorded. In the second and third experiments, respectively, the transmission of spikes and the computation of synaptic currents was disabled; thus, the activity in these two experiments was identical, but the former only drew power to compute neuron states, which allowed the additional power consumed by the latter to be attributed to synaptic activity. It was then possible to derive the key metrics of energy per neuron per simulation tick and energy per synaptic event.

5.4.3 Power consumption

With all spikes transmitted and all synaptic currents computed, the hardware draws an average of 1.95 watts over the observed period. The simulation draws 1.88 watts with synaptic currents disabled, and 1.1 watts with spikes disabled. So, it is possible to estimate the energy required by SpiNNaker to simulate one of the ten thousand Izhikevich neurons in the model for one stimulation tick as

$$\frac{1.096\text{W} \cdot 10^{-3}\text{s}}{10,000 \text{ neurons}} \approx 100\text{nJ / neuron / tick}$$

Of the power consumed, $1.882\text{W} - 1.096\text{W} = 0.786\text{W}$ can be attributed to spikes and synaptic events, of which there are 1,954 and 816,584 respectively in the observed

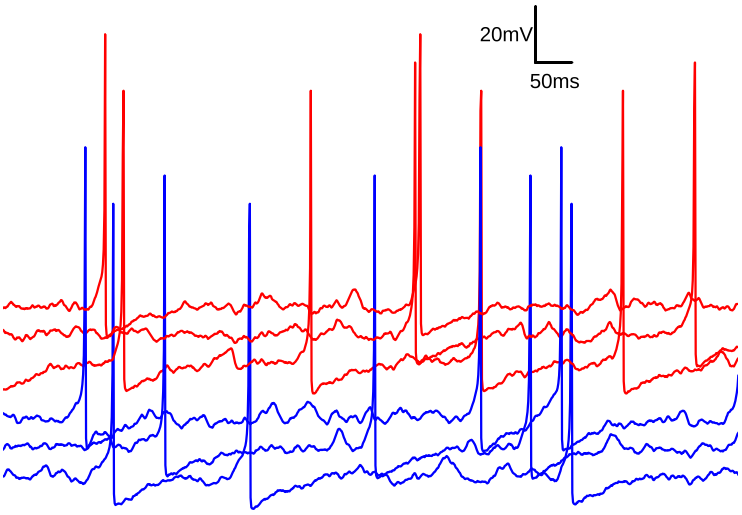


Figure 5.19: Example membrane potentials of neurons in the visual-cortex model.

period. This gives an energy per synaptic event of

$$\frac{0.786\text{W} \cdot 45 \cdot 10^{-3}\text{s}}{816,584 \text{ s.e.}} = 43\text{nJ / s.e.}$$

Figure 5.18 shows power consumption traces in the three experiments, smoothed by a first-order low-pass filter with a time constant of twenty microseconds. Processor activity is clearly driven by the millisecond timer event that prompts neuron states to be computed: when spikes are not transmitted (green trace) this short period of activity is followed by a passive state in which almost one watt of power is consumed between the four chips; in the other two experiments (without weights in red, full simulation in blue) this period is significantly longer as packet and DMA events interrupt to demand attention from the processor, and overall power consumption is increased by ongoing DMA transfers. Full simulation consumes more power than simulation without weights as a result of the greater number of spikes caused by recurrent excitation. The exception is in millisecond 5,100, as a trough in full-simulation firing rates coincides with a random peak in firing rates in the simulation without weights.

5.4.4 Network activity

The cortical model was simulated primarily to understand the power requirements of biologically plausible simulations on SpiNNaker. Nevertheless, recordings of simulation activity were made to demonstrate the model sanity and hardware capabilities. Figure

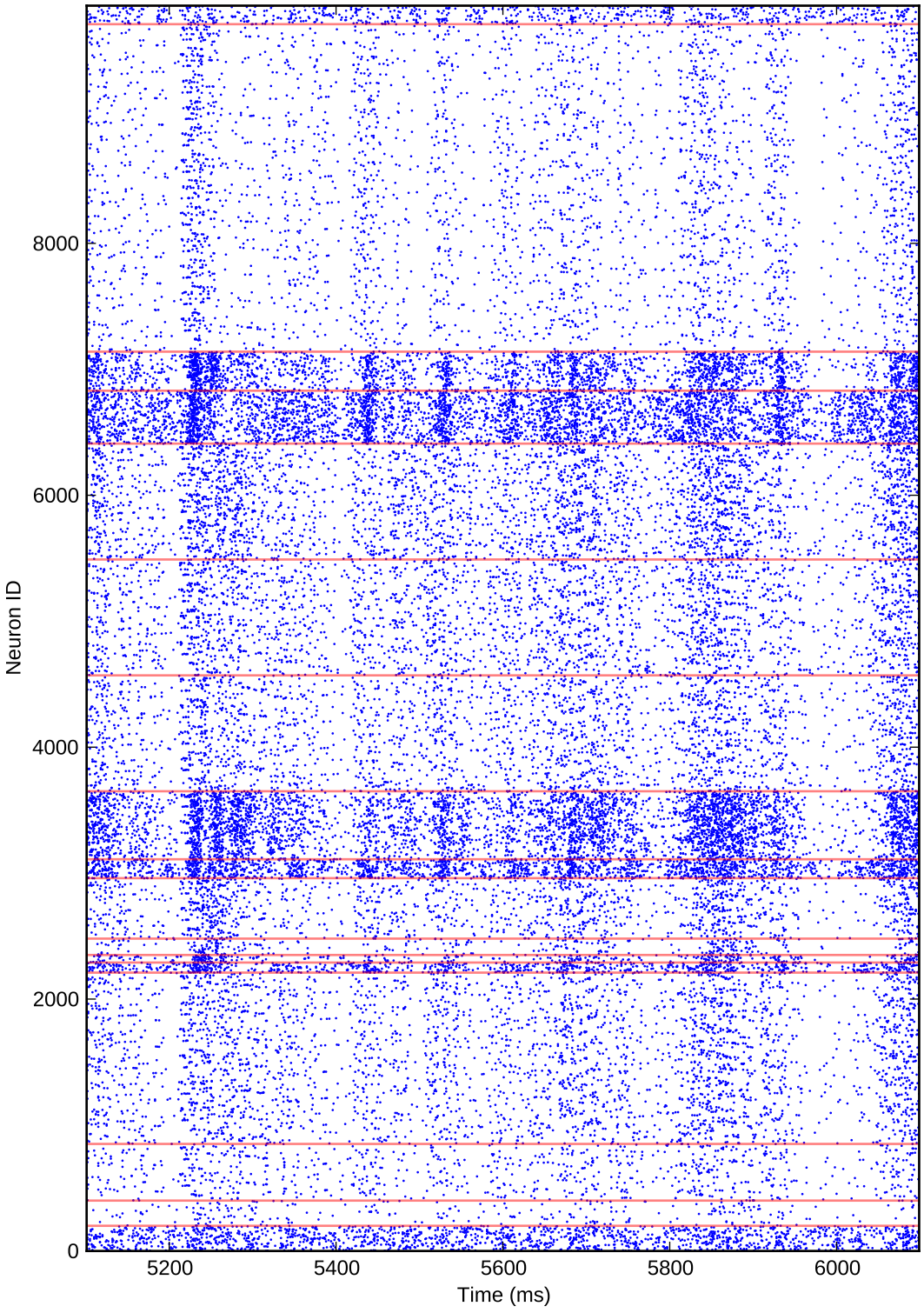


Figure 5.20: Spiking of the visual-cortex model, stratified by population.

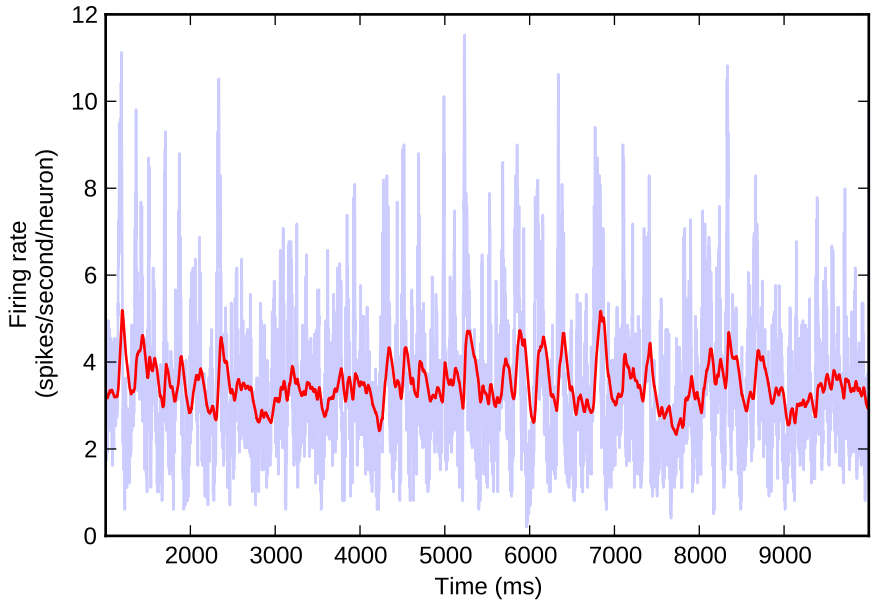
5.19 shows one second of membrane potential traces from three spiny stellate cells (blue) and three basket cells (red) in layer 4. Figure 5.20 displays one second of spike recordings from all cells in the network, stratified by the populations listed in table 5.2; the data clearly show the increased spiking rates of inhibitory cells with respect to excitatory cells, and the weak oscillations that arise from competing excitation and inhibition. Trivial analysis of these data show, as in figure 5.21, that the global-average firing rate oscillates around 4Hz, but that the majority of cells fire at the biologically plausible frequency suggested by Neymotin *et al.* (2011).

5.4.5 Discussion

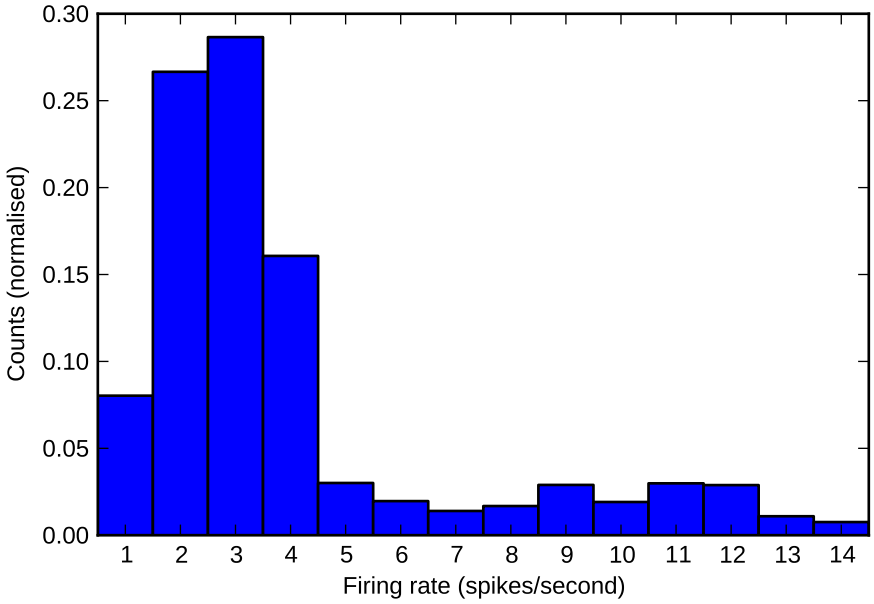
These experiments show that SpiNNaker is capable of simulating complex models drawn from detailed anatomical data and demonstrate the power efficiency of the architecture. Recordings show that four chips consume two watts to simulate, in real-time, 10,000 neurons firing at four hertz into $4{\cdot}10^6$ synapses. These figures break down into key energy metrics of 100nJ per neuron per simulation tick and 43nJ per synaptic event.

The power consumption reported here is lower than expected for the architecture. This may be because of the low clock speeds and light processor loading used in the experiments. Processors were run at 150 megahertz and peripherals at 100; execution at the architectural targets of 200 and 133 megahertz would undoubtedly increase both power consumption and performance. Increasing processor loads from 200 to 1,000 neurons would also increase power consumption, although the significant power overhead of a running processor would then be shared between a greater number of neurons. It should also be noted that power recordings were made directly from the supply rails to the chips and memories; it is reasonable to expect an equal amount of power to be lost in voltage regulation, cooling and other infrastructure (Hölzle and Barroso, 2009).

According to these data, a fifty-thousand-chip, million-processor machine should consume fifty kilowatts to simulate 10^8 neurons. This is five times short of the architectural goal of simulating 10^9 neurons using one hundred kilowatts. Increasing the processor load fivefold would certainly close this gap significantly, as the increased power requirements per processor would be offset by the diminished power overhead per neuron. So, although these data only allow a coarse estimate, it would appear that the power consumption of fabricated SpiNNaker hardware is close to the architectural targets. This implies that real-time simulation of billion-neuron, trillion-synapse models is power-feasible using SpiNNaker. As such, SpiNNaker represents an unprecedented step towards very large-scale simulations of neural tissue.



(a) Instantaneous and smoothed average firing rate of the entire network.



(b) Proportions of neurons at each firing rate.

Figure 5.21: Firing rate statistics of the visual-cortex model.

5.5 Summary

SpiNNaker is a novel architecture designed to simulate large-scale models of neural tissue correctly, quickly and efficiently. The architecture does not contain hardware for floating point operations or for measuring performance or power consumption, so executing simulations and verifying that the fabricated chips meet the design criteria are nontrivial tasks. This chapter has presented a series of experiments of varying novelty to show that SpiNNaker meets or approaches its architectural targets and is therefore an effective large-scale simulation platform.

SpiNNaker correctly simulates neural populations with recurrent projections and external input, although comparing results with reference simulators is not easy. There exists no published standard for the numerical solution of model differential equations, so no two simulators return precisely the same results in even the most simple simulations, as figure 5.4 shows. Nevertheless, section 5.2.3 demonstrates that the outputs of SpiNNaker and NEST do significantly correlate in established comparison metrics.

Performance and power data from prototype hardware suggest that SpiNNaker is a fast, energy-efficient simulator of neural tissue, although comparing results with reference platforms is again difficult. There are no published figures for the power consumption of neural-tissue simulations on any digital hardware other than SpiNNaker. Section 3.2.2 estimates the power requirements of simulation on superscale BlueGene hardware, but it would be inappropriate to compare those figures with the results presented here: the estimate is necessarily very coarse and the scale and parameters of the BlueGene and SpiNNaker simulations differ greatly. Comparisons are also complicated by the presence of the host machine, which contributes significant computational resources to the simulation and consumes significant amounts of power. The effects of the host have been disregarded in the above experiments for two reasons: the proportional contributions and requirements of the host tend to zero as SpiNNaker machines scale towards many thousands of chips; and the host may be a low-power dumb terminal if the hard work of generating simulation data structures and analysing results is done on the SpiNNaker machine itself. With these caveats, the experiments above suggest that SpiNNaker is a promising architecture.

The performance profile of the simulation software shows that each processor may model 256 leaky integrate-and-fire neurons with exponential synaptic currents in real-time while handling 5,000 synaptic events per millisecond. So, the platform meets the performance goals set by Jin *et al.* (2008) to simulate 1,024 Izhikevich neurons with delta synaptic currents, which are approximately one quarter as computationally expensive.

However, Jin *et al.* set a target of handling 10,000 synaptic events per millisecond on the assumption that the processing loop would consist of approximately twenty instructions, whereas it actually contains nearer forty. Nevertheless, the profiling methodology developed here informs both subsequent code optimisations and correct load-balancing of simulations across processors.

The performance profile of the whole software stack shows that SpiNNaker significantly outperforms NEST using similar computational resources to simulate 4,096 neurons. Futhermore, SpiNNaker recruits additional processors and maintains a constant run time as model scale increases; only the load and dump times increase, and these may be ameliorated by on-chip generation and analysis of simulation data. The benefits of rapid simulation are numerous and significant: as run-time grows, so do the notes required to record the purpose of the experiment; the same may be said for the effort required to debug program code and models; the inverse may be said for the number of trials that can be performed in the pursuit of statistically significant results; and, finally, a running simulation represents dead time for a researcher that cannot be usefully filled by switching attentional focus to some other task, due to the overhead involved in doing so. SpiNNaker holds the promise that a simulation of ten seconds of neural activity will take almost exactly ten seconds, and should thereby increase researcher productivity.

The power profile of the hardware shows that it is most energy-efficient. Around 10^4 neurons may be simulated on four chips, which collectively consume around two watts. Assuming that loss through voltage regulation, cooling and other infrastructure brings total power consumption to one watt per chip, fifty-thousand-chip simulations of 10^8 neurons appear readily feasible. This economy is achieved in circuits of a 130 nanometre geometry, and scaling the chips to 45 or 22 nanometre technology may yet offer a tenfold improvement. Digital silicon technology is orders of magnitude more power-hungry than the nervous system, so the energy usage of simulators will become a significant concern as neural tissue models grow to incorporate billions of neurons. This chapter suggests that the SpiNNaker approach of using very many low-power, simple processor cores is appropriate for this scale of simulation.

Hardware	Processing	Memory	OS
Host	AMD Athlon-II X3-445 3 processors 3GHz clock	4GB	Ubuntu 11.10
Multicore	AMD Opteron 6174 48 processors on 4 chips 2.2GHz clock	130GB	CentOS 5.6
SpiNNaker	SpiNNaker chip $16n$ processors on n chips configurable clock speed	32KB code per processor 64KB data per processor 128MB data per chip	N/A

Table 5.5: Hardware platform summary.

Software	Version	Notes
Python	2.7	with NumPy 1.5.1 and SciPy 0.9.0
NEST	2.1	
Brian	1.3.1	
PyNN	0.7	with minor bug fixes on the NEST interface
SpiNNaker	SVN rev. 2269	latest revision of all experiments

Table 5.6: Software component summary.

Chapter 6

Real-Time Million-Synapse Simulation of Cortical Tissue

The nervous system is a powerful and intricate computer, which appears to process information by the genesis and propagation of action potentials amongst billions of neurons. The function of the animal brain may be investigated using models that mimic this mode of information processing, although enormous computational resources are required to do so. The feasible scale and speed of neural-tissue simulations have so far been limited by the parallelism and power disparities between biological and silicon computers, such that it has been impossible to usefully model even million-neuron circuits. The preceding chapters have argued that SpiNNaker addresses these problems, and so portends mega-scale models of neural tissue; this chapter presents simulations towards that end. Data from the biological literature on the rodent somatosensory cortex is tabulated, and experiments are conducted to analytically and empirically establish the correct parameters for a model of the superior barrel column. Basic thalamocortical response transformations, observed both *in vivo* and in simulation, are reproduced in a scale model of the whisker barrel. Multiple barrel columns are instantiated and laterally connected to form hundred-thousand-neuron, seventy-million-synapse models that are simulated in real-time across 360 parallel processors on more than 20 SpiNNaker chips.

6.1 Motivation

There is strong evidence that the action potential is fundamental to rapid information processing in the brain. Spikes traverse distinct pathways in the central nervous system at speeds of many metres per second. Diffusive chemical processes provide homeostatic and developmental support to these tissues, but operate too slowly over large distances to participate in immediate computations. So, it is reasonable to assume that the cortical response to sensory stimuli is primarily a function of action potentials, and that the cortex can therefore be simulated as a network of spiking neuron models. Thorpe *et al.* (1996) show that humans respond to visual cues so rapidly that information must be conveyed to the cortex by a barrage of action potentials along parallel axons in the optic nerve. Simons and Carvell (1989) demonstrate clear correlations between deflections of rodent whiskers and characteristic firing in the corresponding barrels. The functions of both tissues have been captured by models in which spikes are the sole means of interneuron communication (Thorpe *et al.*, 2001; Kyriazi and Simons, 1993).

The rodent barrel cortex is an ideal tissue of which to build models of neural information processing, in that the anatomy and physiology of the whisker system is exceptionally well characterised. Current models of the rodent whisker barrel accurately reproduce the thalamocortical response transformations observed by Simons and Carvell (1989) but use very few neurons (Kyriazi and Simons, 1993) or reduce the activity of whole populations to a single firing-rate state variable (Pinto *et al.*, 2003). In doing so, these models may fail to demonstrate important principles of spike-based information processing that are present *in vivo*. The structural and functional disparities between organic and silicon computers may be to blame for these limitations in simulation studies: the architectures of conventional high-performance computers are quite different from the enormous parallelism, immense energy-efficiency and dense, intricate communication patterns of cortical tissue.

SpiNNaker is designed to emulate the structure and function of neural tissue, with very many low-power processors and an interprocessor communication mechanism inspired by axonal arbours. These adaptations afford sufficient computational power and communication bandwidth to simulate every neuron of the barrel column in real-time on a single four-chip board. However, the true computational power of the whisker system arguably arises from multicolumn processing (Contreras and Llinás, 2001; Civillico and Contreras, 2005; Higley and Contreras, 2005; Civillico and Contreras, 2006) and larger 48-chip boards promise resources to simulate the half-million neurons of the entire barrel cortex. Such models may offer unprecedented insight into how spikes

perform computations in neural tissues and why these tissues are so exquisitely structured. This chapter presents advances in the feasible scale of simulation, with models of increasing scale from one barrel column to nine.

6.2 Simulation methodology

All simulations were run on SpiNNaker, using the hardware described in section 3.2.6 and the software described in chapter 4. Models that spanned fewer than 64 processors were simulated on four-chip circuit boards, and larger models were simulated on 48-chip boards. Each processor was used to host up to 256 neurons, so each n-neuron population was split across $\lceil \frac{n}{256} \rceil$ processors. Multiple four-chip boards were often used in parallel to repeat experiments with varying parameters; the small boards functioned as multiple cheap, energy-efficient simulation accelerators.

All models were specified in PyNN. Upon a call to `run()` PACMAN automatically compiled the necessary data structures on the host and uploaded them to SpiNNaker, which executed the simulations in real-time and returned the results to the host. As such, the mechanisms of compilation and simulation were entirely hidden from the user-level PyNN script and do not warrant repeating. However, some small additions were made to both PACMAN and the on-chip software, which should be described.

A model was created to generate Poisson spike trains on chip. This model is instantiated in PyNN as a `SpikeSourcePoisson` population of size n and is supplied with a parameter dictionary containing a list of (*time*, *probability*) tuples. Each tuple denotes the *probability* of each cell firing in each millisecond from *time* onwards. For example, a list [(100, .05), (300, .025)] denotes no firing from $t = 0$ to $t = 99$, an average of fifty hertz firing from $t = 100$ to $t = 299$, and an average of twenty five hertz firing from $t = 300$ to the end of the simulation. The on-chip model software interprets the parameter list to generate spike trains: in each millisecond t, the firing probability p is determined from the tuple list; then for each neuron i, the processor draws a pseudo-random number x from a uniform distribution over the interval [0, 1) using a software linear feedback shift register (ARM, 1995); if $x < p$, a spike is emitted for neuron i. The spike-train generator may be provided with a seed, or it may generate its own sequence of incrementing seeds over repeated trials. This allows the host to retrigger simulations many times, in order to compile results from many trials, without having to regenerate and reload spike trains.

A trivial addition was made to the activity-recording software on chip in order

to save firing rates of populations. A function `record_fr()` was added to the PyNN `Population` class, which simply sets a flag that prompts the on-chip software to record the number of spikes generated by the population in each millisecond. On a call to `get_fr()` the host retrieves this record, divides each millisecond entry by the number of cells in the population, and multiplies each entry by 10^3 to produce the population-average firing rate in terms of spikes per second. Since downloading and sorting spikes is slow, this method expedites the process of calculating population firing rates.

6.3 Modelling the barrel cortex

The rodent barrel cortex processes mechanical stimuli to facial whiskers. Like the other sensory cortices, the barrel cortex is radially organised into the granular, supragranular and infragranular layers. Axonal projections from the ventral posteromedial nucleus of the thalamus, which convey sensory signals from the whiskers, primarily innervate the granular layer. Broadly, the granular layer innervates the supragranular layers that in turn project down into the infragranular layers. Unlike other sensory cortices, however, the rodent somatosensory cortex exhibits clear lateral organisation of discrete barrels in the granular layer that correspond one-to-one with the whiskers. The notional projection of the barrel column into the supra- and infragranular layers forms a convenient unit for modelling. The average rat column contains 17,560 neurons in a tangential area of 0.12 millimetres squared and a depth of 1.84 millimetres (Meyer *et al.*, 2010a).

The barrel column is well defined anatomically, compared to the visual cortex for example, and the functional relationship between whiskers and barrels is clear. Yet, like any other volume of cortex, each barrel column is profoundly complex. The boundaries between layers are fuzzy and there are no clear septa between columns in the supra- and infragranular layers. The neurons of these hazily defined populations are heterogeneous in terms of morphology, physiology and synaptic connectivity; Oberlaender *et al.* (2011) and Binzegger *et al.* (2004) alone identify numerous classes of excitatory and inhibitory cells in each layer based on these criteria. Synapses adapt to spiking activity on a range of time scales, so observed synaptic currents vary markedly between experiments *in vivo* and *in vitro* and with different stimulation protocols.

This complexity presents significant challenges to modelling and simulation: the literature does not contain data for every parameter of a spiking model of the barrel column; detailed, many-parameter models inevitably require arbitrary tuning to reproduce biological function; and verbatim models of tissue, if at all possible, are no better

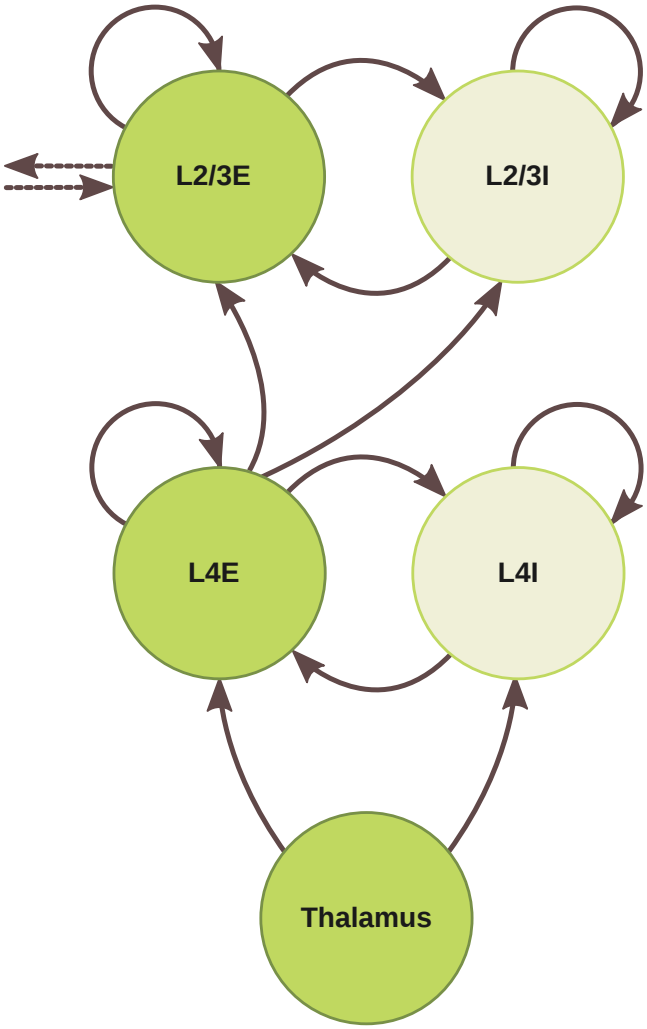


Figure 6.1: Architecture of the superior barrel column model.

for explaining fundamental computational principles than the living cortex itself. One solution is to form hypotheses about which features of the tissue are computationally significant and to then disregard all others.

As such, specific attention was paid to information processing in the superior layers of the barrel column. A model was constructed to examine the stimulus-response of the granular-layer barrel and the propagation of activity between columns in the supragranular layers. The populations and projections of the model, henceforth called the *superior barrel column*, are depicted in figure 6.1. The supragranular layers L2 and L3 were represented by a single layer L2/3, as they commonly are in the literature; supragranular layer 1 is very cell-sparse and was ignored. Each layer contained one excitatory and one inhibitory population, which were connected both recurrently and

Population	Exci.	Inhi.	Source
Layer 2/3	4,507	795	Meyer *et al.* (2010a)
Layer 4	3,471	613	ibid.
Thalamus	285	—	Oberlaender *et al.* (2011)

Table 6.1: Barrel column neuron counts.

to one another. The thalamus fed simulated whisker signals to both populations of the granular layer, and the excitatory granular neurons relayed signals to the supragranular layers. In models of multiple columns, supragranular populations formed lateral projections between one another. Synapses from the thalamus to L2/3 were not modelled, despite evidence that they exist in rodents (Meyer *et al.*, 2010b). Nonuniform distributions of synaptic weights, connection probabilities and connectivity motifs were disregarded, despite evidence that they may be significant (Tomm, 2012; Perin *et al.*, 2011). Data from different rodent species were combined into one model due to the scarcity of literature on any particular species.

Parameter	Unit	L2/3E	L4E
E_L	mV	-72	-66
V_{reset}	mV	-72	-66
V_Θ	mV	-40	-40
τ_m	ms	30	35
R_m	MΩ	190	300
Refrac.	ms	10	10
τ_{s_e}	ms	5	5
τ_{s_i}	ms	15	15

Table 6.2: Barrel column neuron physiology (Lefort *et al.*, 2009).

The literature provides data on the size of populations, the probability of projections and the physiology of both. Table 6.1 lists population sizes for the thalamus and the cortex, which were found by Oberlaender *et al.* (2011) and Meyer *et al.* (2010a) using automated counts of NeuN-positive cells in slices and the assumption that 0.15× of cortical neurons are inhibitory. Table 6.2 presents physiological properties of the model neurons, according to results of sharp electrode recordings performed by Lefort *et al.* (2009); synaptic-current time constants were taken from Kyriazi and Simons (1993) and

		Pre					
		L2/3E	L2/3I	L4E	L4I	Thal.	**Sources**
Post	L2/3E	.12[1]	.55[2]	.13[1]	—	—	[1]Lefort *et al.* (2009)
	L2/3I	.40[2]	.60[2]	?	—	—	[2]Avermann *et al.* (2012)
	L4E	—	—	.24[1]	.21[3]	.67[4]	[3]Sun *et al.* (2006)
	L4I	—	—	.13[3]	?	.67[4]	[4] Meyer *et al.* (2010b)
	Thal.	—	—	—	—	—	

Table 6.3: Barrel column connectivity.

Sun *et al.* (2006). Equivalent data for inhibitory neuron physiology does not exist. Table 6.3 lists projection probabilities according to four studies: Lefort *et al.* (2009) investigate excitatory connectivity in all layers with paired electrode experiments *in vitro*; Avermann *et al.* (2012) use optogenetic techniques *in vitro* to map excitatory and inhibitory connectivity in the supragranular layers; Sun *et al.* (2006) use paired electrodes to test the same connectivity *in vitro* in the granular layer, albeit with very small ($n \approx 70$ cell pairs) sample sizes; and Meyer *et al.* (2010b) count thalamocortical synapses in 3D reconstructions of fixed tissue. These authors variously describe synapse physiology in terms of postsynaptic potential latencies, amplitudes and distributions but these data cannot be usefully employed in models, as later argued. The nature or existence of the projection from granular excitatory neurons to supragranular inhibitory neurons is unknown and not modelled.

The thalamic and cortical response to ramp-and-hold whisker stimuli is well characterised *in vivo* and in simulation by Simons and Carvell (1989), Kyriazi and Simons (1993) and Pinto *et al.* (2003). The whisker barrel exhibits four response transformations on thalamic input, namely: thalamic neurons have greater levels of spontaneous spiking than cortical excitatory neurons; cortical excitatory neurons respond with different numbers of spikes to the onset and offset of whisker deflection, whereas thalamic neurons do not; cortical excitatory neurons respond weakly to deflection of an adjacent whisker, unlike thalamic neurons; and the response of excitatory neurons to principal whisker deflection is suppressed if it is immediately preceded by deflection of an adjacent whisker. The simulated barrel was expected to reproduce the first and second of these response transformations in order to validate the model. Each thalamic neuron was set to generate Poisson spike trains at a mean rate of six hertz (Bruno and Sakmann, 2006) in the absence of stimuli. Pinto *et al.* show that the thalamic response to onset and

offset stimuli differs in terms of firing synchrony or onset rate, and the firing rate of the thalamic spike trains was varied to mimic this effect: onset stimuli triggered a stimulus *triangle* rising from six hertz to thirty hertz in five milliseconds and then decaying back to six hertz in thirty milliseconds; offset stimuli triggered a stimulus triangle of equal amplitude and opposite rise and decay times. These stimuli are visible in figure 6.8.

6.4 Construction of biologically plausible models

In a model composed of excitatory and inhibitory neural populations, firing-rate dynamics are largely determined by the synaptic connectivity of the network (Brunel, 2000). Should the number and strength of excitatory synapses outweigh those of inhibitory synapses, for example, recurrent excitation will dominate and the network will become hyperactive. Should inhibition dominate, the network will become inactive. Between these two pathological states a great variety of rich dynamics are possible. In the animal nervous system the health and computational abilities of neural tissue are presumably maintained by some homeostatic mechanism. This mechanism has yet to be described in detail, so modellers must use either empirical or analytical methods to find the parameters of synaptic connectivity that elicit biologically plausible activity. The former method typically entails time-consuming guesswork that does not lead to a reasoned understanding of the model. The latter method, developed by Brunel (2000), is demonstrably effective but it is currently only applicable to simple models. A series of experiments were conducted in an attempt to extend the latter methodology to address more detailed models of neural tissue, in order to lay foundations for a biologically plausible model of the superior barrel column.

6.4.1 Asynchronous networks

Brunel considers a model comprising N_e excitatory and N_i inhibitory leaky integrate-and-fire neurons, driven by some external input, where every neuron receives a synapse from every other neuron with equal probability p. The number of synapses formed within the network is $p(N_e + N_i)^2$ so the ratio of excitatory to inhibitory synapses is simply N_e/N_i. Given some arbitrary excitatory synaptic weight w_e, a significant relationship to the inhibitory weight w_i is described by

$$w_i = bw_e\frac{N_e}{N_i} \tag{6.1}$$

where b is a balance coefficient. Brunel plots model firing rate as a function of b and shows that excitation drives hyperactivity when b is less than one, that inhibition causes hypoactivity when b is greater than one, and that the two forces counterbalance when b is exactly one. Furthermore, the balanced network ($b = 1$) exhibits irregular, asynchronous firing akin to that observed in cortical networks.

The whisker barrel is an ideal model with which to reproduce the Brunel experiments, since it has been described as two populations, one excitatory and one inhibitory, driven by a sole thalamic input (Kyriazi and Simons, 1993; Pinto *et al.*, 2003). However, in contrast to the Brunel model, the barrel exhibits significant heterogeneity in neuron and synapse physiology and projection probabilities: excitatory neurons have longer membrane time constants than inhibitory (Simons and Carvell, 1989), inhibitory synapses have longer current-decay time constants than excitatory (Sun *et al.*, 2006), and projection probabilities between the two neuron types vary significantly (Lefort *et al.*, 2009; Avermann *et al.*, 2012). Equation 6.1 implicitly describes excitatory-to-inhibitory balance in terms of the type, number and efficacy of synaptic events that occur in the network; when these parameters are not equal for the excitatory and inhibitory populations, the equation must be extended.

The charge imparted by an exponential postsynaptic current with amplitude w and time constant τ_s is

$$\int_0^\infty w \exp(-t/\tau_s)\, dt = w\tau_s$$

so currents in a network with different excitatory and inhibitory synaptic time constants balance when

$$N_e \tau_{se} w_e = N_i \tau_{si} w_i$$

The average number of excitatory synapses in a network with heterogeneous projection probabilities is

$$N_e(p_{ee}N_e + p_{ei}N_i)$$

where p_{ee} and p_{ei} are the probabilities of excitatory-to-excitatory and excitatory-to-inhibitory projections; the number of inhibitory synapses is determined likewise. So, synaptic currents balance in a network with varying projection probabilities when

$$N_e(p_{ee}N_e + p_{ei}N_i)\tau_{se} w_e = N_i(p_{ii}N_i + p_{ie}N_e)\tau_{si} w_i$$

The firing rate of the leaky integrate-and-fire neuron is inversely proportional to the

membrane time constant τ_m so the Brunel-equation extensions are completed with

$$\frac{1}{\tau_{me}} N_e(p_{ee}N_e + p_{ei}N_i)\tau_{se}w_e = \frac{1}{\tau_{mi}} N_i(p_{ii}N_i + p_{ie}N_e)\tau_{si}w_i$$

The balance coefficient can be reintroduced, and the equation rearranged for w_i

$$w_i = bw_e \frac{N_e(p_{ee}N_e + p_{ei}N_i)\tau_{se}\tau_{mi}}{N_i(p_{ii}N_i + p_{ie}N_e)\tau_{si}\tau_{me}} \tag{6.2}$$

which reduces to equation 6.1 if time constants and projection probabilities are equal.

6.4.2 Methodology

A model of the whisker barrel was simulated to reproduce the Brunel experiments regarding balance of excitatory and inhibitory currents. Thalamic and cortical neurons were simulated as Poisson spike trains and leaky integrate-and-fire neurons respectively. Population sizes and neuron physiologies were determined by tables 6.1 and 6.2. The remaining parameters were varied in successive experiments, from a model with homogeneous population physiologies and projection probabilities, to one that incorporated the heterogeneity evident in the literature. Each experiment sought to find the relationship between the balance coefficient b and the average firing rate of the excitatory neurons. The excitatory synaptic weight was fixed at 0.1 nanoamps and the inhibitory weight was determined by equation 6.2, with b varying from 0.1 to 10 in successive trials. In each trial the model was instantiated and loaded once and then simulated ten times with varying seeds for the Poisson spike source; the average excitatory firing rate during each one-second simulation was recorded, and the ultimate result of the trial was the mean and standard deviation of these ten numbers.

Three model *forms* were simulated. In all forms, excitatory and inhibitory neuron parameters were as listed for layer 4 cells in table 6.2. In the *homogeneous form*, all intracortical projection probabilities were 0.1 and inhibitory synaptic time constants were the same as excitatory, as in the original Brunel experiments. In the τ_s *form*, synaptic time constants varied true to table 6.2. In the *P form*, probabilities of the four intracortical projections varied in accordance with table 6.3. The forms thereby progressed from a simple model of the barrel to a biologically plausible one, although varying membrane time constants were not incorporated for reasons described below.

Each form was simulated under two different thalamocortical projection probabilities. A *plausible* projection with probability 0.67 (Meyer *et al.*, 2010b) was initially used.

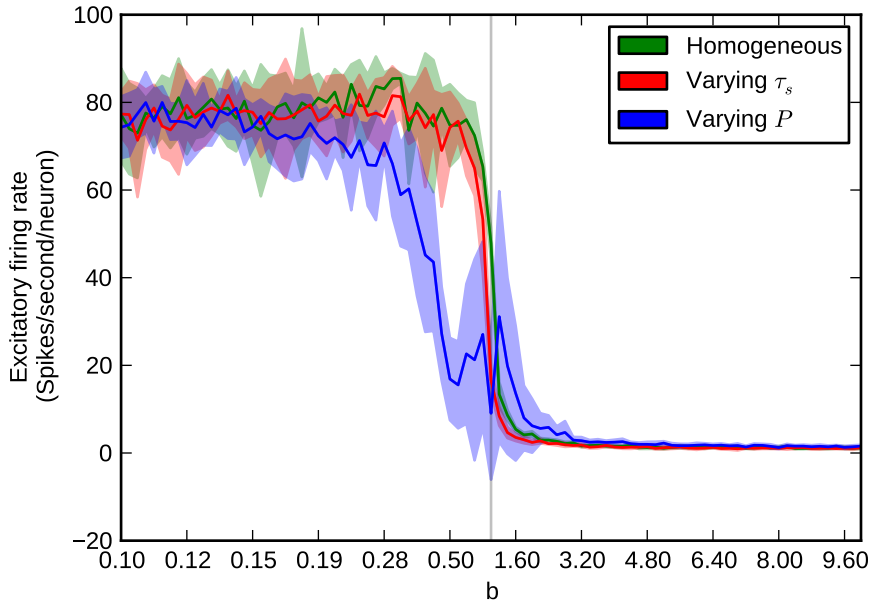


Figure 6.2: Barrel firing rate under plausible input as a function of balance.

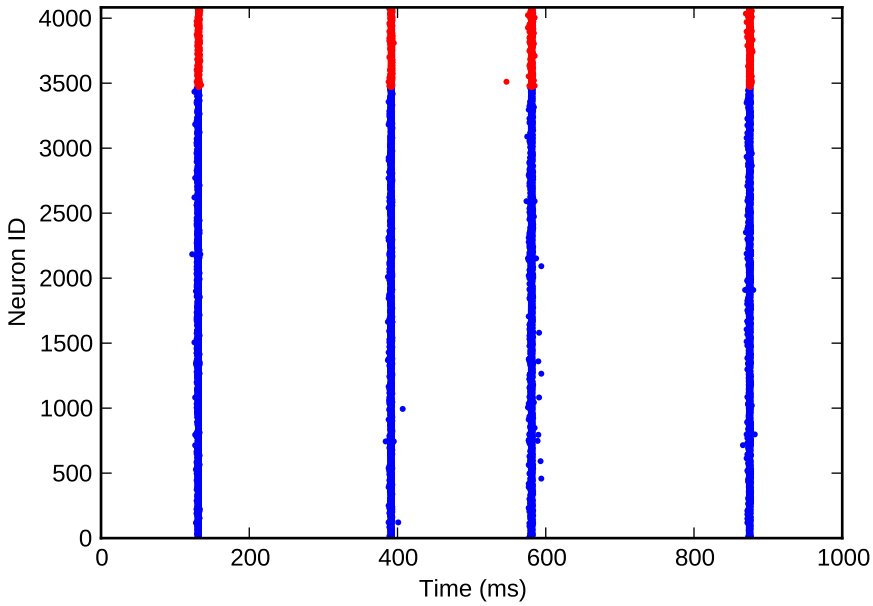


Figure 6.3: Hypersynchronised firing in the barrel under plausible input.

When this was found to produce hypersynchronous firing in the barrel, an *implausible* projection with probability 0.25 was used. In both cases, synaptic weights were chosen to be the minimum required to elicit firing in a model without intracortical projections.

6.4.3 Results

Figure 6.2 shows excitatory firing rate in the barrel as a function of the balance coefficient, under stimulus from the plausible thalamic projection. The homogeneous and τ_s forms follow the expected curve precisely: as b sweeps from less than to greater than one, the firing rate transitions from near-maximum to near-minimum through a tight sigmoid curve centred upon $b = 1$. The P form shows unexpected deviation and variance, in that the firing rate curve drops off before $b = 1$ and the standard deviation around that point is large.

Figure 6.3 shows that the spike times of the P form at $b = 1$ are clearly not irregular and asynchronous, but rather are hypersynchronised. The number of hypersynchronous bursts varied from trial to trial (data not shown) which accounts for the large standard deviation in firing rate. The other forms suffered similar pathological firing patterns (data not shown) and it was hypothesised that the highly divergent thalamic connectivity was responsible.

Figure 6.4 shows the results of the same experiment performed with the implausible thalamic projection. Again, the homogeneous- and τ_s forms follow the expected curve, with notably smaller standard deviations than before. And again, the P form deviates unexpectedly, in that the firing rate curve is nonmonotonic and peaks at $b = 1$. To investigate the nature of this peak, ten barrel models were instantiated with the parameter $b = 1$ and each was simulated ten times.

Figure 6.5 shows the distribution of average excitatory and inhibitory firing rates across these hundred trials. The firing rates are biologically plausible in most trials but are hyperactive in a small number. This makes the P form unsuitable for reliable simulations of the barrel column, and further experiments with varying membrane potential time constants would likely be futile. It appears that the extended Brunel equations do not adequately describe the competing effects of excitatory and inhibitory synaptic events in P form.

The extended equation does correctly describe the behaviour of the τ_s form, and may be used as a basis for an empirical search for the correct parameters of the barrel model. Figure 6.4 suggests that the τ_s form under implausible input should fire at fifty hertz in an irregular asynchronous regime at $b = 1$. This result was confirmed by sim-

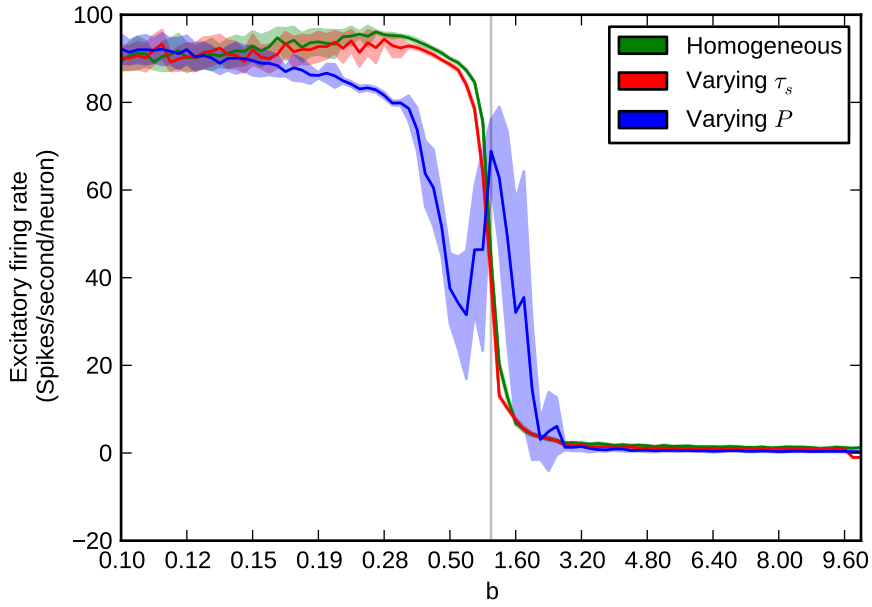


Figure 6.4: Barrel firing rate under implausible input as a function of balance.

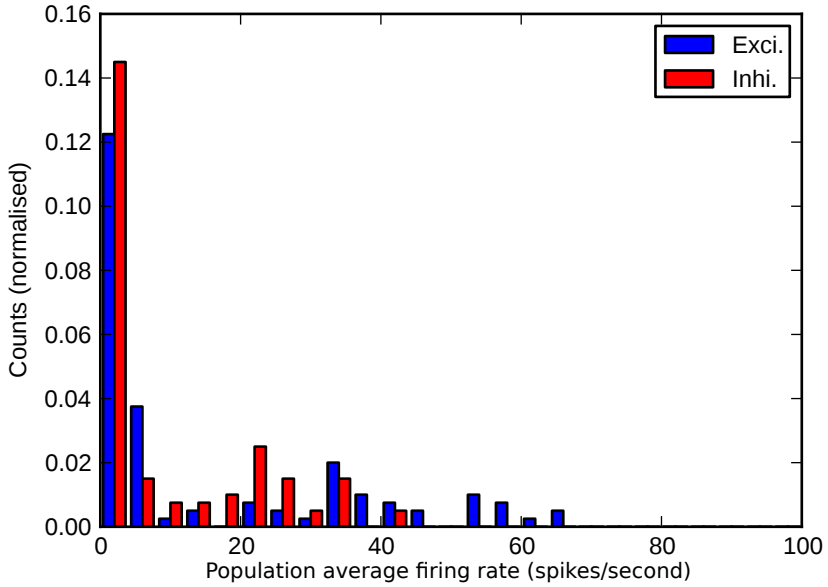


Figure 6.5: Distribution of the trial-average firing rates of the barrel P form.

ulating such a model (data not shown) and the experiment was repeated speculatively with a coefficient of 1.5 on the inhibitory-to-excitatory weight to produce the result in figure 6.6.

The biological plausibility of this ultimate network activity is debatable. The excitatory neurons appear to fire irregularly, in that the average coefficient of variation of interspike intervals is 0.99 ± 0.37, suggesting plausible Poisson-like activity (Softky and Koch, 1993). Also, the firing rates of the thalamic, excitatory and inhibition populations and approximately correct in relation to one another. However, the distribution of average firing rates of excitatory cells is not log-normal, as expected by Roxin (2011); rather, firing rates are exponentially distributed, as shown in figure 6.7. It appears that significant further work would be required to produce a network with both balanced, irregular spiking and the correct firing-rate distribution.

6.4.4 Discussion

Neural tissue is so intricate that even small models are necessarily complex in topology and dynamics, which makes dynamical-systems analysis of stability or stimulus-response extremely difficult. This is a principal problem in the field of neural-tissue simulation, in that no methodology currently exists to understand the fundamental properties of the models under examination here. In the absence of analytical tech-

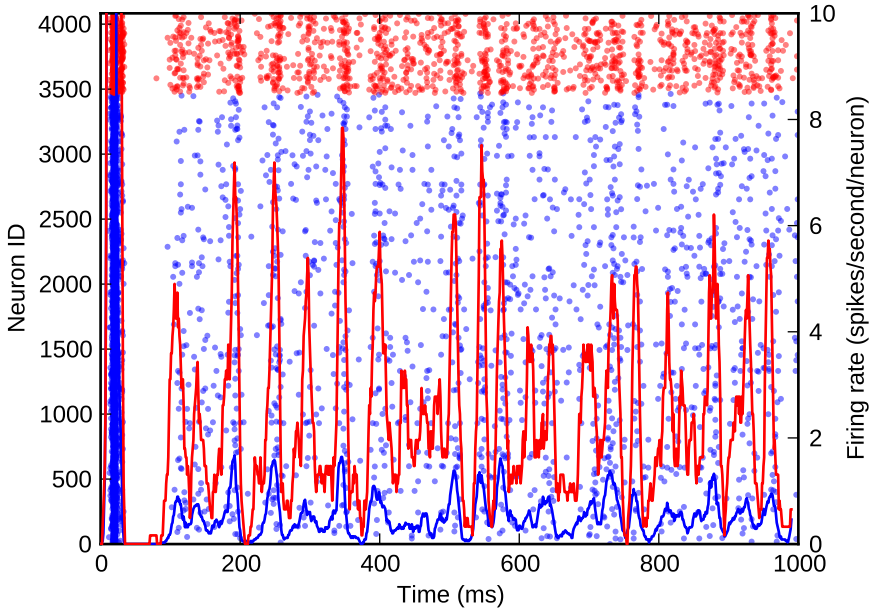


Figure 6.6: Spiking of the ultimate barrel model.

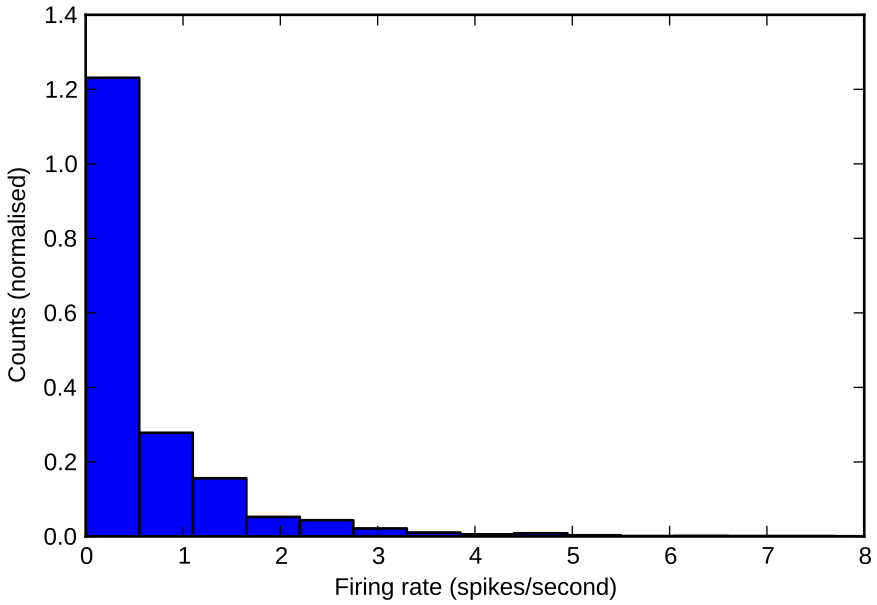


Figure 6.7: Proportions of neurons at each firing rate in the ultimate barrel model.

niques one must revert to empirical methods of construction and tuning, which are extremely time-consuming and may not elucidate the important properties and parameters of the ultimate model.

This dissertation was conceived, in part, with the expectation that neuroanatomical data could be transcribed directly into models of the cortex, which may then be used to investigate the higher functions of the brain. This expectation is resolutely false. In an early experiment, a model of the barrel column was built that included all available data on the anatomy and physiology of the tissue, down to the weight distributions of each synaptic projection; the model completely failed to function like its biological inspiration and many weeks were spent tuning it to do so. A more reasoned approach to designing simulations of the cortex is clearly necessary, and this section is the modest result of efforts to that end.

Brunel (2000) deals comprehensively with the dynamics of homogeneous networks of spiking model neurons. A simple equation describes the stability of the asynchronous state found at the balance of excitation and inhibition in such networks, and extensions were made here to the equation to consider the effects of heterogeneity in synapse physiology and projection probabilities. These extensions appear to be invalid, in that they do not adequately describe the balance of forces required to establish the desired irregular, asynchronous firing in a model of excitatory and inhibitory neurons. As

projection probabilities between populations were varied, the firing rate of the network became an unpredictable function of the balance coefficient b.

The Brunel balance equations are based upon the assumption that the desired firing results from equality in the number and efficacy of excitatory and inhibitory postsynaptic currents in a network. The number of excitatory (inhibitory) currents that occur in some period is the product of the firing rate of the excitatory (inhibitory) neurons and the number of synapses that they make. The former is clearly a dynamical parameter and it appears that those dynamics are limited in homogeneous networks, such that the Brunel equation adequately describes the balance of excitation and inhibition. In heterogeneous networks where recurrent excitation is significantly stronger than other projections, as in the whisker barrel, larger excitatory dynamics may undermine the balance equations. This would explain why the P form has a tendency to become hyperactive. A dynamical-systems approach, perhaps drawing upon the analyses by Pinto *et al.* (2003) and Izhikevich (2007), to further work may be more fruitful.

6.5 Large-scale simulation of the barrel cortex

The barrel cortex is an ideal tissue upon which to base large-scale simulations. Although biologically plausible models of the tissue are apparently difficult to build, abstract representations of the barrel are useful units with which to construct scalable networks. Where one barrel column is simulated, many may be simulated in parallel with lateral connections between supragranular populations. Such models are useful for exploring the capabilities of SpiNNaker. A series of experiments were conducted to reproduce the basic function of a single barrel and to observe the operation and performance of large-scale barrel-field models on SpiNNaker.

6.5.1 Barrel-field models

A model of the superior barrel column was constructed according to the anatomical data, modelling assumptions and findings of the model-building experiments above. Figure 6.1 shows the overall architecture of the model. Thalamic and cortical neurons were simulated as Poisson spike trains and leaky integrate-and-fire neurons respectively. Population sizes and neuron physiologies were determined by tables 6.1 and 6.2. Within each layer, projection probabilities were set to 0.1, the excitatory synaptic weight was set to 0.1 nanoamps, and the inhibitory synaptic weight was determined automatically according to equation 6.2; the inhibitory-to-excitatory weight was also

multiplied by 1.5, as discussed above. Between layers, projection parameters were tuned to elicit the desired activity: the projection probability from the excitatory granular neurons to both supragranular populations was set to 0.1, and the synaptic weight of both projections was set to 0.2 nanoamps. Between columns, lateral projections were formed between excitatory populations of the supragranular layers, with probability 0.1 and synaptic weight of 0.1. To reiterate the stimulus parameters: thalamic neurons fired at a baseline rate of six hertz, and responded to simulated ramp-and-hold whisker deflections with firing-rate triangles of thirty-hertz amplitude and onset and offset times of five and thirty milliseconds; these are apparent in figure 6.8. Following from the Brunel experiments, which demonstrated that building models directly from anatomical data is inordinately difficult, the pretence of biological plausibility was abandoned in favour of demonstrating the large-scale simulation capabilities of SpiNNaker.

Each superior column contained approximately 9,000 neurons. To explore the capability of SpiNNaker, a chain of five columns and a grid of nine were simulated. In both cases, one column was stimulated and the signal propagation through the supragranular layers was observed. However, firstly an experiment was conducted to reproduce basic thalamocortical response transformations in the barrel model.

6.5.2 Thalamocortical response transformations

The granular layer was simulated alone under thalamic input to examine thalamocortical response transformations in the model. The two transformations under consideration were the lower firing rate of excitatory neurons with respect to thalamic neurons and the differential response of excitatory neurons to onset and offset stimuli. Whisker deflection onset and offset was modelled by thalamic firing rate triangles, as discussed above. A stimulus battery was defined as one whisker deflection onset and one offset separated by 150 milliseconds, preceded by a 500-millisecond rest period. Ten barrel models were instantiated in succession and 25 stimulus batteries were delivered to each.

Figure 6.8 shows the spike times resulting from one stimulus battery and the average firing rates across all of the 25 batteries to all ten model instances. For aesthetic reasons the stimulus battery is centred in the figure. The barrel model clearly reproduces the two response transformations under consideration: excitatory neurons fire with asynchronous irregularity at frequencies much lower than thalamic neurons, and onset stimuli elicit greater excitatory responses than offsets. The firing rates of the excitatory and inhibitory neurons are not in precise agreement with biological observations (Neymotin *et al.*, 2011) but they are correct in relative proportion to one another.

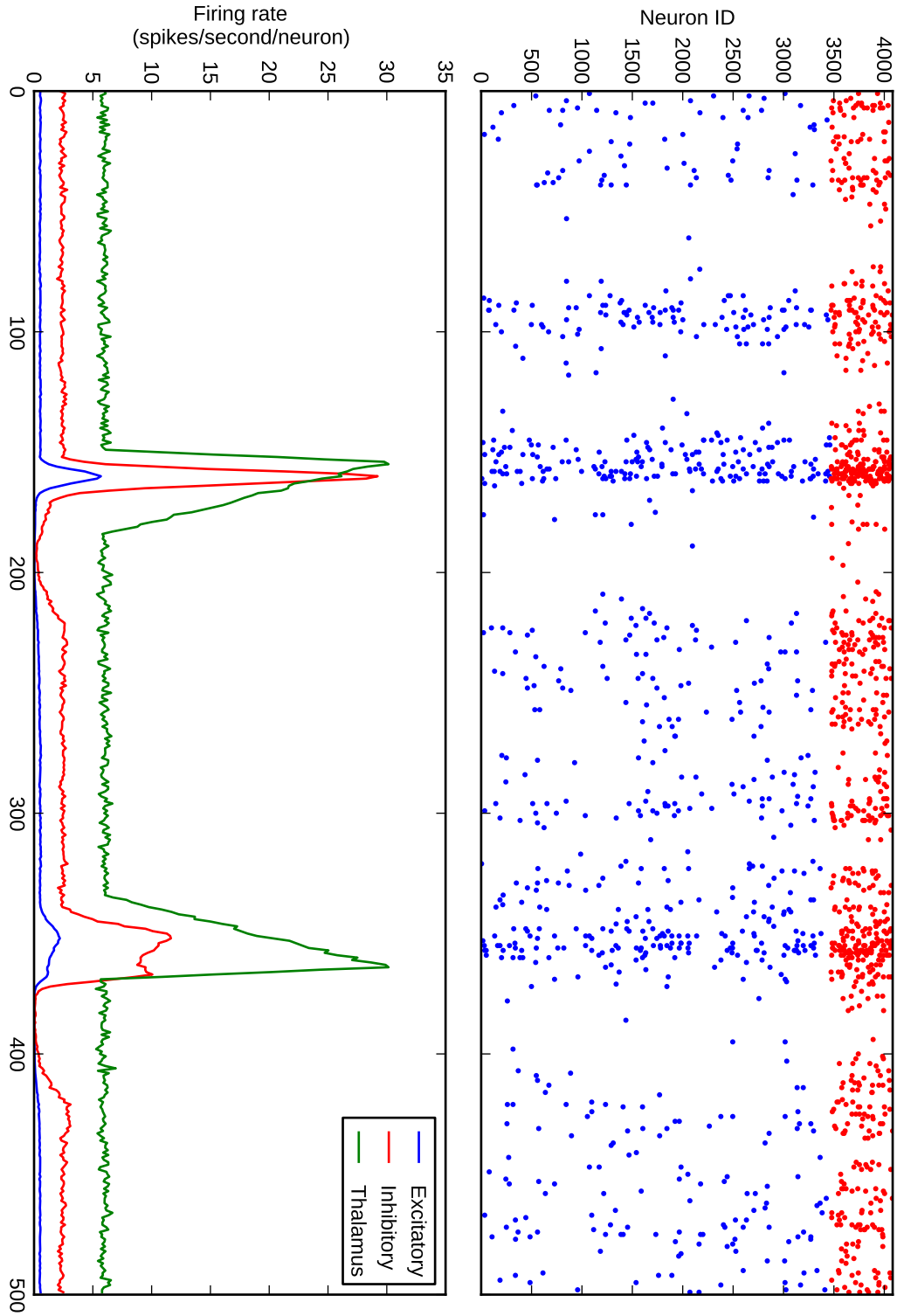


Figure 6.8: Thalamocortical response transformation in the whisker barrel.

Pinto *et al.* (2003) hypothesise that the thalamocortical response transformation to whisker deflection is a function of inhibitory response: when stimulus onset is slow, inhibitory firing increases gradually with the stimulus and suppresses excitation; when stimulus onset is fast, both cell classes respond simultaneously and excitation rises to greater levels before inhibition has an effect. This certainly seems to be the case in figure 6.8 but Pinto *et al.* argue that it is a consequence of the shorter membrane time constants of inhibitory neurons; the time constants of both cell classes are equal in this experiment, which suggests otherwise. The response transformation may more generally result from a greater influence of inhibitory neurons on the network; in this case it is the increased inhibitory-to-excitatory synaptic weight, whereas in the case of Pinto *et al.* it may be the different excitatory and inhibitory membrane time constants.

6.5.3 Barrel-column chain

Civillico and Contreras (2006) observe that activity in the supragranular layers propagates between columns via a lateral spread of excitatory axons. A chain of barrel columns was constructed in order to simulate such activity and to develop the large-scale modelling capability of SpiNNaker. Five columns were instantiated and laterally connected in the supragranular layer with reciprocal projections. Thus, the model consisted of approximately fifty thousand neurons and fifty million synapses simulated across 200 processors on 13 chips. A stimulus battery was defined as five whisker oscillations at ten hertz, preceded by a 500-millisecond rest period; the battery was delivered 25 times to the leftmost column while the others received only baseline stimulus, and the firing rates of the thalamic and excitatory cortical populations were recorded.

Figure 6.9 shows the average peristimulus time histogram in spikes per second per neuron across all stimulus-presentations, again centred upon the stimulus battery. The top, middle and bottom panels represent the supragranular, granular and thalamic populations respectively, and the five traces in each panel from bottom to top represent the five columns from left to right; note the varying y-scale bars on each panel. The thalamic stimulus is visible in the bottom trace of the bottom panel, and the effect on the corresponding granular neurons is evident in the middle panel; note that the thalamic and granular neurons of the other columns show no response. The activated granular layer relays signals to the corresponding supragranular layer, from which the firing clearly propagates along the chain of columns. It would be bold to presume that this experiment precisely reproduces some function of the barrel cortex, but it does serve to demonstrate simulated activity spreading across a massively-parallel machine.

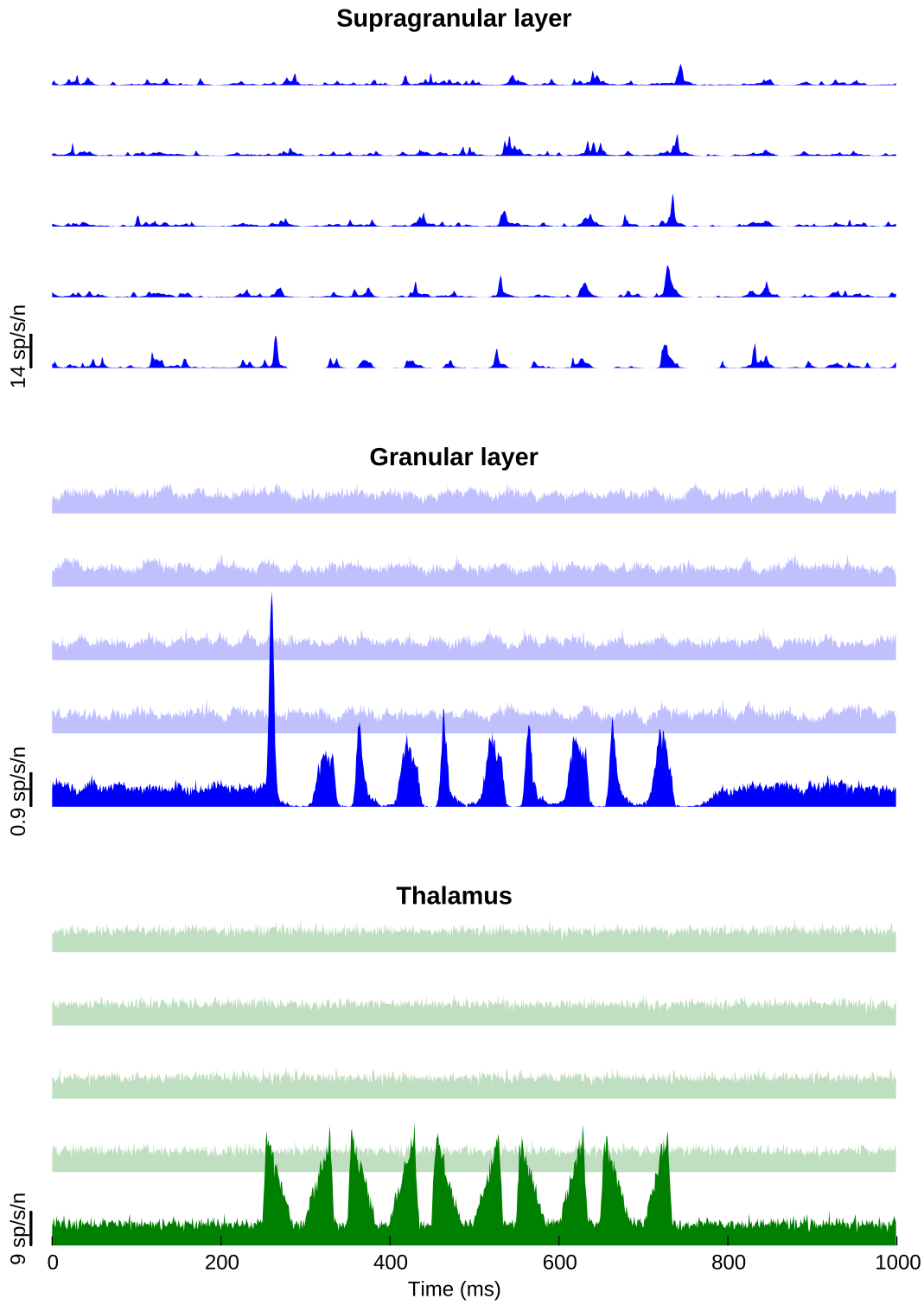


Figure 6.9: Signal propagation through a chain of barrel columns.

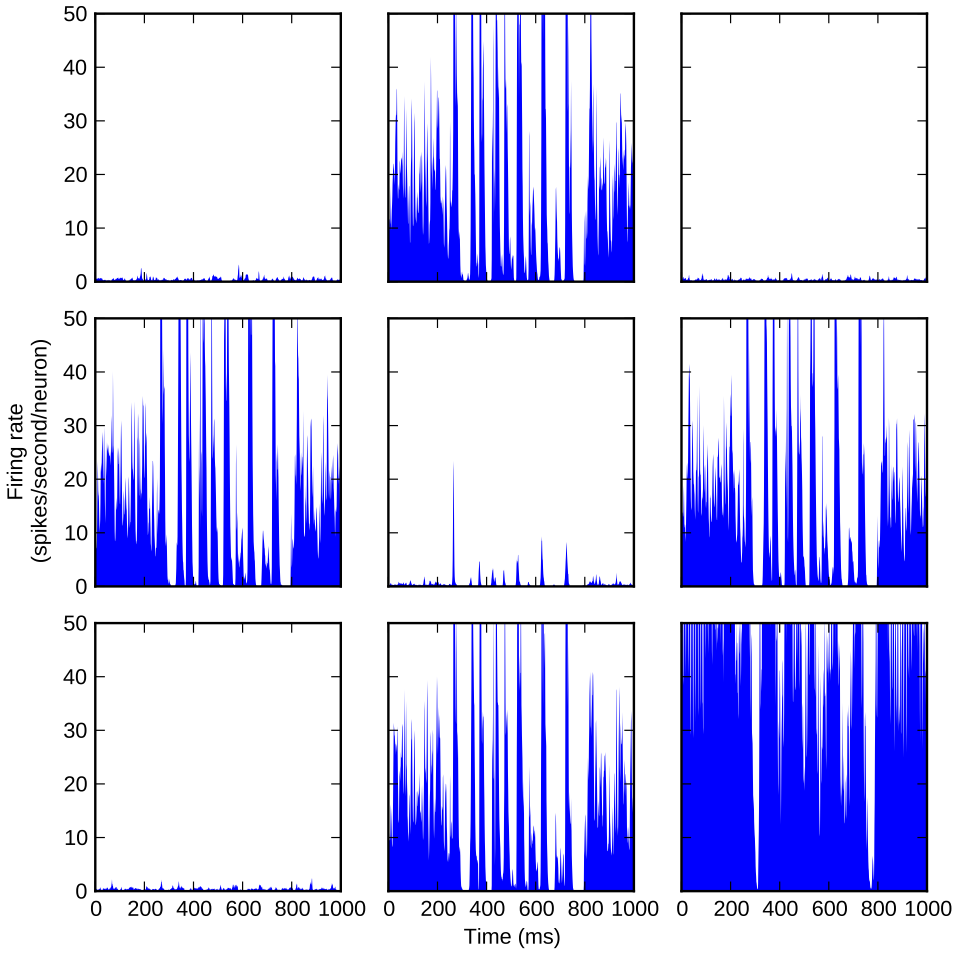


Figure 6.10: Activity in the supragranular layer of a grid of barrel columns.

6.5.4 Barrel-column grid

By way of further demonstration, a grid of barrel columns was simulated on SpiNNaker. Nine columns were instantiated in a three-by-three grid, and unidirectional projections were formed as shown in figure 6.10; a full grid of reciprocal connections was initially attempted but uncontrollable hyperactivity resulted. The same stimulus as before was presented to the central barrel, and the spread of activity was observed.

Figure 6.10 shows activity in the supragranular layer emanating from the central column. The Northerly, Southerly, Easterly and Westerly columns show increased, synchronised activity as a function of their dual inputs from their barrels and the central column; this effect is yet more apparent in the South-Easterly population, which receives two lateral projections. The South-Westerly, North-Westerly and North-Easterly columns exhibit only baseline activity, driven by their corresponding granular popula-

tions. The activity is rather implausible and the model is no basis for investigation into neurocomputational function. However, the simulation is significant as a raw demonstration of feasible scale: the model consists of approximately ninety thousand neurons and seventy million synapses, simulated across 360 processors on 23 chips. And the feasible scale may yet be increased with improvements to the supporting software:

Figure 6.11 shows the execution profile of the simulation, including time spent in host-side programs. The run time consists, as expected, of the 25 one-second stimulus presentations plus one leading and one following second to allow the model and the machine respectively to settle. Each of the other profile components significantly outweighs this cost. The synaptic intricacy of the circuit requires the host to spend more than a minute building half a gigabyte of data structures, which then take more than twenty minutes to load. The 25-second firing-rate trace takes 150 milliseconds to retrieve from each processor, so the dump time is almost a minute in total. Clearly then, SpiNNaker itself is an effective simulator of large-scale models of neural tissue, although improvements to the host-side tools and loading protocols are required.

6.6 Summary

The elementary function of the cortex is arguably best understood in the rodent whisker system. The responses of individual barrels to whisker stimuli has been examined *in vivo* (Simons and Carvell, 1989) and reproduced in simulation (Kyriazi and Simons, 1993; Pinto *et al.*, 2003) to a greater and more plausible degree than any other tissue. This chapter demonstrates that SpiNNaker can model this classic barrel activity and that of multiple barrel columns in parallel, so to further investigate the response transformations of multiple-whisker stimuli in the somatosensory cortex (Civillico and Contreras, 2006).

The anatomy and physiology of the barrel is well characterised, but constructing biologically plausible models of the tissue is nontrivial. Models with recurrent synaptic projections are extremely sensitive to parameters of excitability and connectivity, which makes establishing a balance of excitation and inhibition by empirical methods difficult. Section 6.4 considers analytical techniques developed by Brunel (2000) for finding such a balance in simple models, and attempts to extend them to accommodate some of the complexity of the cortex. These extensions appear to be invalid, in that they do not engender the desired irregular, asynchronous firing in a model of the barrel. However, the original Brunel equation does inform such activity, and forms the basis for full-scale

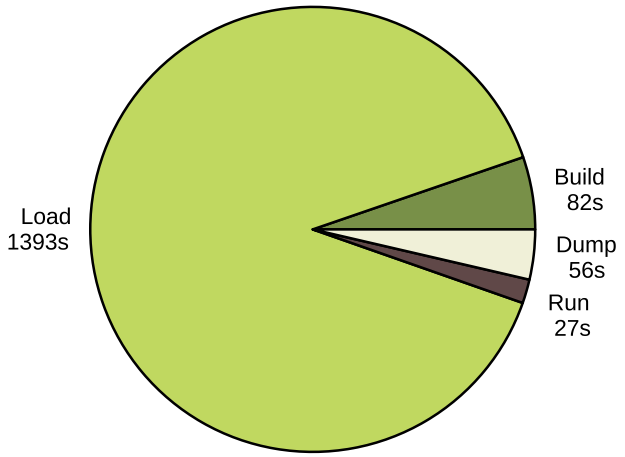


Figure 6.11: Execution-time profile of the barrel grid model.

but somewhat implausible models of the superior layers of the barrel column.

Initial experiments with these models show that the barrel reproduces thalamocortical response transformations observed *in vivo*. Pinto *et al.* (2003) observe that the thalamus represents whisker deflection onset and offset as barrages of spikes with equal amplitude and varying synchrony, between which the cortex can discriminate. Simulated spike trains of this form were delivered to a barrel model, and excitatory cells were shown to respond more keenly to the onset of whisker deflection than the offset. This is an important validation of both full-scale modelling of the barrel cortex and the SpiNNaker simulator.

Civillico and Contreras (2006) observe that activity in the superior layers propagates between columns via a lateral spread of excitatory axons. Chains and grids of barrel columns were constructed in order to simulate approximately such activity and to examine the large-scale modelling capability of SpiNNaker. Figure 6.9 shows the effect of thalamic stimulus upon one barrel and the spread of this activity into and across the supragranular populations. Figure 6.10 shows similar effects in a grid of barrel columns, as simulated in real-time on 360 processors across 23 SpiNNaker chips. The scale and execution-speed of this experiment represents a significant advancement in the technology for neural tissue simulation.

The current limitation to further scaling beyond simulations of 10^8 synapses on SpiNNaker is simply the build and load time of the data structures. This problem may

be trivially addressed by on-board generation of connectivity data; processors aboard SpiNNaker may build their own synapses, and do so in parallel. Considering a projection of probability p from a source population of size n to a destination population of size m, a processor must draw nm uniformly distributed random numbers from the interval [0, 1] and generate a synapse for each one that is less than p; if m is greater than the single-processor capacity of 256 neurons then the destination population is split across multiple processors that operate in parallel. Build time for one projection may be estimated from approximate instruction counts of each operation: uniform pseudo-random numbers may be drawn from [0, 1) on the ARM architecture in five instructions (ARM, 1995); the loop guards over the ranges $i = 1..n$ and $j = 1..m$ may take another five; and the operations to write out each synapse that hits may take a further five. So, the time required is approximately

$$t(n, p) = \frac{256n(10\text{C} + 5p\text{C})}{20 \cdot 10^6\text{C/s}}$$

where C denotes processor cycles and the denominator represents an operating frequency of 200 megahertz. Since each processor may generate synapses in parallel, build time for an entire machine is equal to the sum of the projection-build-times of the most loaded processor; this is in contrast to host-side building and loading, the time for which grows linearly with the number of synapses in the model. The processors that simulate the supragranular excitatory population of the South-Easterly column in figure 6.10 receive five projections, each of probability 0.1, from a total of 18,000 source neurons, so they build their synapses in

$$t(18{,}000,\ 0.1) \approx 0.25 \text{ seconds.}$$

All other processors complete in equal or lesser time. So, assuming that the time for loading projection descriptions to processors is trivial, this approach entirely ameliorates the cost of building and loading synapse data and thereby further accelerates simulation of large-scale models.

Chapter 7

Conclusions

The brain is a computer of incomparable power, efficiency and complexity, the understanding of which is a foremost concern in science, medicine and engineering. Simulation of neural tissue is a promising but tentative methodology to this end. This dissertation has presented significant advances in technology for simulating large volumes of the cortex, but has also identified fundamental problems with such modelling methods. It appears that fast, energy-efficient simulation is feasible on the SpiNNaker architecture, but that construction and operation of biologically plausible models is a difficult open problem.

SpiNNaker is a massively parallel, real-time computer architecture that employs very many low-power processors and novel communications hardware in mimicry of the structure and function of the nervous system. This design affords extreme computational power and efficiency to the simulation of neural tissue, but presents significant challenges to programmability. Parallel and real-time programming techniques are notoriously difficult, and are no less so when combined. The principal contributions of this dissertation are large-scale, biologically inspired simulations of the cerebral cortex that demonstrate the efficacy of SpiNNaker, and the design, implementation and testing of the software upon which these demonstrations are based.

Real-time simulation is driven on SpiNNaker by events from which computational tasks must follow. In each processor, periodic timer events prompt numerical solution of neuron states and packet-received events trigger calculation of synaptic currents. The handling of these events and consequent scheduling of tasks must be carefully orchestrated if correct and real-time operation is to be achieved. The application programming interface (API) and run-time kernel (ARK) presented in chapter 4 serve this purpose. The API allows arbitrary user functions to be registered at certain priorities

with the ARK, and the latter calls these functions in priority order in response to each event. In this paradigm, neuron and synapse models may be specified for simulation in standard C code, and the execution-order of tasks arising from concurrent events may be controlled. Both of these properties are essential: the ideal abstraction of the neuron is unknown, so any simulator of neural tissue must be able to model arbitrary voltage and current dynamics; and temporal control over the scheduling of tasks must be carefully exercised if none is to unduly dominate the processor. The performance and power consumption of simulation code built atop the API is examined in chapter 5, and results show that SpiNNaker is an effective architecture. Processors are capable of simulating between 256 and 1024 neurons in real-time while handling 5,000 synaptic events per millisecond, and the ARK and API routines incur very little overhead. Sixteen-processor chips draw approximately one watt each during simulation, so million-processor models are comfortably power-feasible. A single chip outperforms NEST running on an equivalently powerful machine, and SpiNNaker simulations parallelise and scale easily across many chips.

Models of neural tissue are specified for SpiNNaker simulation in PyNN as *populations* of neurons and synaptic *projections* between the former. PyNN is a Python-based programming interface that is simulator-agnostic, and is therefore useful for hiding the complexities of massively parallel hardware and for comparing the results produced by different simulators. The latter point is demonstrated in chapter 5, which shows that SpiNNaker correctly reproduces results from the established simulators Brian and NEST. The former point is exhibited in chapters 5 and 6 with models of the cerebral cortex drawn from the biological literature; in these examples PyNN is used to design and execute simulations of the cat and rat cortices with no regard for the architecture of the underlying simulator. Thus, the implementation of PyNN for SpiNNaker completes a software stack that allows researchers in computational neuroscience to exploit high-performance hardware to simulate models of neural tissue without any knowledge of parallel programming.

The rodent whisker system is an ideal candidate for biologically plausible modelling. The structure and function of the barrel cortex is relatively well understood, so simulations of this area may be built from and validated against biological data. Chapter 2 presents a comprehensive survey of the relevant literature, and the tools that are discussed in chapters 4 and 5 ultimately drive the simulations, presented in chapter 6, that aim to reproduce the function of the barrel cortex in large-scale, data-based models. These experiments achieve mixed success. The essential function of the barrel can

be reproduced on SpiNNaker, and a model of nine columns comprising $7{\cdot}10^7$ synapses may be simulated in real-time across 360 processors; these results represent a significant advance in the feasible scale and speed of neural-tissue simulations. However, the barrel-column model used in both experiments is rather implausible.

Networks of spiking neurons are dynamical systems that are extremely sensitive to parameters of excitability and connectivity. The parameters that maintain irregular, asynchronous firing *in vivo* are not captured by current physiological techniques. Consequently, any model constructed entirely from data in the literature will simply fail to function as expected, usually suffering from hyperactivity, hypersynchrony, hypoactivity, unresponsiveness to stimuli or some other pathological state. This is evident in the models in chapters 5 and 6, which were conceived as tools for investigating cortical function but were, after necessary parameter tweaking that destroyed biological plausibility, repurposed as demonstrations of SpiNNaker. The approaches to finding model parameters for correct function are either empirical or analytical; the former entails time-consuming guesswork that does not lead to a reasoned understanding of the model, and the latter is limited to rather simple networks with homogeneous neuron parameters and projection probabilities. Chapter 6 presents developments to the existing analytical methodology that attempt to account for the heterogeneity in cortical tissue, and puts forward a hypothesis about why these developments are not successful. So, ultimately, the primary contributions of this dissertation are large, fast simulations of coarse cortical models on novel hardware and software.

.

Simulation of neural tissue remains a tentative methodology in neuroscience because of uncertainty about the plausibility and potential of modelling techniques. Abstract models that are amenable to quantitative analysis are clearly vital if the essential function of the nervous system is to be explained, but the appropriate degree of abstraction is contended. This dissertation has assumed throughout that spiking neurons are fundamental to cortical function, but numerous studies have argued the same of, for example, postsynaptic potential integration in complex dendritic trees or aggregated firing frequencies of entire populations. The arguments against either end of the spectrum are that too much detail in a model obfuscates its fundamental properties, and that too little severs the connection between the model and the modelled. In favour of the former argument, chapter 6 shows that classic response transformations can be reproduced in an implausible model that does not incorporate heterogeneous projection

probabilities between excitatory and inhibitory neurons. In favour of the latter argument, one might ask why the cortex contains such heterogeneity at all if its function may be reproduced in far simpler models of tissue.

There are also practical issues in simulation studies that warrant diversion from the philosophical. The principal challenges in cortical-scale simulations are power consumption and interprocessor communication: neural tissue is many orders of magnitude more energy-efficient than silicon technology and conventional communications hardware is ill suited to conveying multicast action-potential pulses. SpiNNaker is carefully adapted to these requirements and demonstrates significant success in both areas, but these achievements come at the sacrifice of auxiliary hardware and synaptic plasticity algorithms.

SpiNNaker does not contain hardware for boot-up, data transfer, simulation control, debugging, profiling or floating-point computations. Some of these tasks may be handled entirely by software, some would be significantly enhanced by additional hardware, and some are simply impossible without; examples include, respectively, onboard generation of data structures that avoids transmission of large binaries, profiling of simulation programs at run-time, and single-step debugging of simulations across multiple processors. Auxiliary hardware may increase chip-design cost and reduce performance, but certain additions may enormously accelerate the pace of software development and debugging. The correct balance between performance and programmability is not obvious, but SpiNNaker may err slightly in favour of the former.

The hardware that SpiNNaker does contain is optimised for a model of simulation in which spikes innervate synapses that in turn drive neural activity. Processors compute membrane potentials and synaptic currents, and the communications and memory systems serve solely to retrieve data with which to feed these computations. The modification and storage of synaptic weights, required for learning through activity-dependent synaptic plasticity, are therefore cumbersome operations on SpiNNaker. In particular, processors are insufficiently powerful to compute weight modifications for every inbound synapse, and the latency of the off-chip memory is such that spike-induced read-modify-writes on scattered synapses are impractically slow. However, SpiNNaker is not unusual in this regard: synaptic plasticity algorithms are inherently expensive and present significant challenges to million-synapse simulations on any architecture.

.

Computer engineering has enjoyed apparently inexorable progress since the development of the integrated circuit. The future offers ever greater computational resources

and iterations of the SpiNNaker architecture that should improve performance and resolve the technical problems noted here. The volumes of neural tissue that can be feasibly modelled will certainly continue to grow. But the philosophical concerns with the methodology will require specific attention: what is the correct level of abstraction for models of neural tissue, how might plausible, functional models be constructed, and what are these models intended to explain?

Neural computation takes place on a continuum of spatial and temporal scales, and modelling techniques will vary according to the particular scale of interest. This dissertation has considered the immediate cortical response to sensory stimuli in terms of spikes amongst networks of neurons. The level of abstraction here is neurons and synapses with simple dynamics that, respectively, generate and respond to action potentials. These components are composed into networks according to an assemblage of biological data, and are empirically tuned to produce the desired activity. The resulting models bear little semblance to the tissue, and are consequently limited in explanatory power as to why, say, certain neuron types have particular physiologies or why they connect to one another with the synaptic properties observed *in vivo*. Developments in both biology and mathematics are required to advance from this position.

Current models of the cortex are necessarily incomplete because no extant dataset describes the complete composition and connectivity of any cortical area. If plausible models of the cortex are to be devised, the target tissue must be completely mapped with a consistent set of techniques and the resulting data must be presented in machine-readable form. However, any degree of abstraction precludes a pure transliteration from data to model; an analytical understanding of the dynamics of spiking neural networks is required if biological observations are to be translated into valid simulations.

Should these developments take place, simulation methodology may produce significant advances in the understanding of the brain. In the process of building valid models of neural tissue from biological data, with analytical descriptions of the desired dynamics, we may come to understand the relationships between neural structure and function that engender the enormous capabilities of the cortex. In simulating such models, we may reproduce, observe, permute and ultimately explain the fundamental computations that take place within.

Bibliography

Mehdi Adibi, Mathew E. Diamond, and Ehsan Arabzadeh. Behavioral study of whisker-mediated vibration sensation in rats. *Proceedings of the National Academy of Sciences*, 109:971–976, 2012.

Jose-Manuel Alonso and Harvey A. Swadlow. Thalamocortical Specificity and the Synthesis of Sensory Cortical Receptive Fields. *Journal of Neurophysiology*, 94:26–32, 2005.

AMD. Online technical specification, 2013. URL `http://products.amd.com/pages/OpteronCPUDetail.aspx?id=643`. Retrieved on the 23rd of May 2013.

Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *American Federation of Information Processing Societies, Spring Joint Computer Conference*, pages 483–485, 1967.

Rajagopal Ananthanarayanan, Steven K. Esser, Horst D. Simon, and Dharmendra S. Mohda. The Cat is Out of the Bag: Cortical Simulations with 10^9 Neurons, 10^{13} Synapses. In *High Performance Computing, Networking, Storage and Analysis, Conference on*, pages 1–12, 2009.

Ehsan Arabzadeh, Rasmus S. Petersen, and Matthew E. Diamond. Encoding of Whisker Vibration by Rat Barrel Cortex Neurons: Implications for Texture Discrimination. *The Journal of Neuroscience*, 23:9146–9154, 2003.

Ehsan Arabzadeh, Stefano Panzeri, and Matthew E. Diamond. Whisker Information Carried by Rat Barrel Cortex Neurons. *The Journal of Neuroscience*, 24:6011–6020, 2004.

ARM. Programming techniques. Technical Report ARM DUI 0021A, 1995.

David Atwell and Simon Laughlin. An Energy Budget for Signaling in the Grey Matter of the Brain. *Journal of Cerebral Blood Flow and Metabolism*, 21:1133–1145, 2001.

Michael Avermann, Christian Tomm, Celine Mateo, Wulfram Gerstner, and Carl C. H. Petersen. Microcircuits of excitatory and inhibitory neurons in layer 2/3 of mouse barrel cortex. *Journal of Neurophysiology*, 107:3116–3134, 2012.

Mohammad A. Bhuiyan, Vivek K. Pallipuram, and Melissa C. Smith. Acceleration of Spiking Neural Networks in Emerging Multi-core and GPU Architectures. In *Parallel Distributed Processing, International Symposium on*, pages 1–8, 2010.

T. Binzegger, R. J. Douglas, and K. A. C. Martin. Topology and dynamics of the canonical circuit of cat V1. *Neural Networks*, 22:1071–1078, 2009.

Tom Binzegger, Rodney J. Douglas, and Kevan A. C. Martin. A Quantitative Map of the Circuit of Cat Primary Visual Cortex. *The Journal of Neuroscience*, 24:8441–8453, 2004.

Tom Binzegger, Rodney J. Douglas, and Kevan A. C. Martin. Sterotypical Bouton Clustering of Individual Neurons in Cat Primary Visual Cortex. *The Journal of Neuroscience*, 27:12242–12254, 2007.

Laurens W. J. Bosman, Arthur R. Houweling, Cullen B. Owens, Nouk Tanke, Olesya T. Shevchouk, Negah Rahmati, Wouter H. T. Teunissen, Chiheng Ju, Wei Gong, Sebastiaan K. E. Koekkoek, and Chris I. De Zeeuw. Anatomical pathways involved in generating and sensing rhythmic whisker movements. *Frontiers in Integrative Neuroscience*, 5:1–28, 2011.

brainmaps.org. Online image, 2013. URL `http://brainmaps.org/ajax-viewer.php?datid=1&sname=0851`. Retrieved on the 16th of May 2013.

A. Braitenberg and A. Schüz. *Anatomy of the Cortex: Statistics and Geometry*. Springer-Verlag, 1991.

Romain Brette. Exact Simulation of Integrate-and-Fire Models with Synaptic Conductances. *Neural Computation*, 18:2004–2027, 2006.

Romain Brette and Wolfram Gerstner. Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. *Journal of Neurophysiology*, 94: 3637–3642, 2005.

Romain Brette, Michelle Rudolph, Ted Carnevale, Michael Hines, David Beeman, James Bower, Markus Diesmann, Abigail Morrison, Philip Goodman, Frederick Harris,

Milind Zirpe, Thomas Natschläger, Dejan Pecevski, Bard Ermentrout, Mikael Djurfeldt, Anders Lansner, Olivier Rochel, Thierry Vieville, Eilif Muller, Andrew Davison, Sami El Boustani, and Alain Destexhe. Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23:349–398, 2007.

Nicolas Brunel. Dynamics of Sparsely Connected Networks of Excitatory and Inhibitory Spiking Neurons. *Journal of Computational Neuroscience*, 8:183–208, 2000.

Nicolas Brunel and Vincent Hakim. Fast global oscillations in networks of integrate-and-fire neurons with low firing rates. *Neural Computation*, 11:1621–1671, 1999.

Nicolas Brunel and Simone Sergi. Firing frequency of leaky intergrate-and-fire neurons with synaptic current dynamics. *Journal of Theoretical Biology*, 195:87–95, 1998.

Nicolas Brunel and Xiao-Jing Wang. What Determines the Frequency of Fast Network Oscillations With Irregular Neural Discharges? I. Synaptic Dynamics and Excitation-Inhibition Balance. *Journal of Neurophysiology*, 90:415–430, 2003.

Randy M. Bruno and Bert Sakmann. Cortex Is Driven by Weak but Synchronously Active Thalamocortical Synapses. *Science*, 312:1622–1627, 2006.

Andrew Cassidy, Andreas G. Andreou, and Julius Georgiou. Design of a One Million Neuron Single FPGA Neuromorphic System for Real-Time Multimodal Scene Analysis. In *Information Sciences and Systems, Annual Conference on*, pages 1–6, 2011.

E. F. Civillico and D. Contreras. Comparison of Responses to Electrical Stimulation and Whisker Deflection Using Two Different Voltage-sensitive Dyes in Mouse Barrel Cortex in Vivo. *The Journal of Membrane Biology*, 208:171–182, 2005.

Eugene F. Civillico and Diego Contreras. Integration of Evoked Responses in Supragranular Cortex Studied With Optical Recordings In Vivo. *Journal of Neurophysiology*, 96:336–351, 2006.

Diego Contreras and Rodolfo Llinás. Voltage-Sensitive Dye Imaging of Neocortical Spatiotemporal Dynamics to Afferent Activation Frequency. *Journal of Neuroscience*, 21: 9403–9413, 2001.

Nuno Maçarico da Costa and Kevan A. C. Martin. Whose cortical column would that be? *Frontiers in Neuroanatomy*, 4:1–10, 2010.

Sergio Davies, Javier Navaridas, Francesco Galluppi, and Steve Furber. Population-Based Routing in the SpiNNaker Neuromorphic Architecture. In *Neural Networks, International Joint Conference on*, pages 1932–1939, 2012.

Andrew P. Davison, Daniel Brüderle, Jochen M. Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. PyNN: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, 2:1–10, 2009.

Peter Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2001.

C. P. J. de Kock, R. M. Bruno, H. Spors, and B. Sakmann. Layer- and cell-type-specific suprathreshold stimulus representation in rat primary somatosensory cortex. *The Journal of Physiology*, 581:139–154, 2007.

A. Destexhe, M. Rudolph, J.-M. Fellous, and T. J. Sejnowski. Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. *Neuroscience*, 107: 13–24, 2001.

Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, Franck Cappello, Barbara Chapman, Xuebin Chi, Alok Choudhary, Sudip Dosanjh, Thom Dunning, Sandro Fiore, Al Geist, Bill Gropp, Robert Harrison, Mark Hereld, Michael Heroux, Adolfy Hoisie, Koh Hotta, Zhong Jin, Yutaka Ishikawa, Fred Johnson, Sanjay Kale, Richard Kenway, David Keyes, Bill Kramer, Jesus Labarta, Alain Lichnewsky, Thomas Lippert, Bob Lucas, Barney Maccabe, Satoshi Matsuoka, Paul Messina, Peter Michielse, Bernd Mohr, Matthias S. Mueller, Wolfgang E. Nagel, Hiroshi Nakashima, Michael E. Papka, Dan Reed, Mitsuhisa Sato, Ed Seidel, John Shalf, David Skinner, Marc Snir, Thomas Sterling, Rick Stevens, Fred Streitz, Bob Sugar, Shinji Sumimoto, William Tang, John Taylor, Rajeev Thakur, Anne Trefethen, Mateo Valero, Aad van der Steen, Jeffrey Vetter, Peg Williams, Robert Wisniewski, and Kathy Yelick. The International Exascale Software Roadmap. *International Journal of High Performance Computer Applications*, 25:3–60, 2011.

Rodney J. Douglas, Kevan A. C. Martin, and David Whitteridge. A Canonical Microcircuit for Neocortex. *Neural Computation*, 1:480–488, 1989.

Kayvon Fatahalian and Mike Houston. A Closer Look at GPUs. *Communications of the ACM*, 51:50–57, 2008.

Andreas K. Fidjeland and Murray P. Shanahan. Accelerated Simulation of Spiking Neural Networks Using GPUs. In *Neural Networks, International Joint Conference on*, pages 1–8, 2010.

Steve Furber and Andrew Brown. Biologically-Inspired Massively-Parallel Architectures — computing beyond a million processors. In *Application of Concurrency to System Design, International Conference on*, pages 1–8, 2009.

Steve Furber and Steve Temple. Neural systems engineering. *Journal of the Royal Society Interface*, 4:193–206, 2006.

Edgar Gabriel, Graham. E Fagg, George Bosilca, Thara Angskun, Jack. J Dongarra, Jeffrey. M Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph. H Castain, David. J Daniel, Richard. L Graham, and Timothy. S Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *European PVM/MPI Users' Group Conference*, pages 97–104, 2004.

Francesco Galluppi, Alexander Rast, Sergio Davies, and Steve Furber. A General-Purpose Model Translation System for a Universal Neural Chip. In *Neural Information Processing, International Conference on*, pages 58–65, 2010.

Francesco Galluppi, Sergio Davies, Alexander D. Rast, Thomas Sharp, Luis A. Plana, and Steve B. Furber. A Hierarchical Configuration System for a Massively Parallel Neural Hardware Platform. In *Computing Frontiers, International Conference on*, pages 183–192, 2012.

A. Gara, M. A. Blumrich, D. Chen, G. L. T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49:195–212, 2005.

Marc-Oliver Gewaltig and Markus Diesmann. NEST (NEural Simulation Tool). *Scholarpedia*, 2:1430, 2007.

Dan Goodman and Romain Brette. Brian: a simulator for spiking neural networks in Python. *Frontiers in Neuroinformatics*, 2:1–10, 2008.

Anirudh Gupta, Yun Wang, and Henry Markram. Organizing Principles for a Diversity of GABAergic Interneurons and Synapses in the Neocortex. *Science*, 287:273–278, 2000.

Stefan Haeusler and Wolfgang Maass. A Statistical Analysis of Information-Processing Properties of Lamina-Specific Cortical Microcircuit Models. *Cerebral Cortex*, 17:149–162, 2006.

Stefan Haeusler, Klaus Schuch, and Wolfgang Maass. Motif distribution, dynamical properties, and computational performance of two data-based cortical microcircuit templates. *Journal of Physiology - Paris*, 103:73–87, 2009.

Bing Han and Tarek M. Taha. Neuromorphic Models on a GPGPU Cluster. In *Neural Networks, International Joint Conference on*, pages 1–8, 2010.

A. Hanuschkin, S. Kunkel, M. Helias, A. Morrison, and M. Diesmann. A general and efficient method for incorporating precise spike times in globally time-driven simulations. *Frontiers in Neuroinformatics*, 4:1–19, 2010.

R. A. Haring, R. Bellofatto, A. A. Bright, P. G. Crumley, M. B. Dombrowa, S. M. Douskey, M. R. Ellavsky, B. Gopalsamy, D. Hoenicke, T. A. Liebsch, J. A. Marcella, and M. Ohmacht. Blue Gene/L compute chip: Control, test, and bring-up infrastructure. *IBM Journal of Research and Development*, 49:289–301, 2005.

M. Helmstaedter, C.P.J. de Kock, D. Feldmeyer, R. M. Bruno, and B. Sakmann. Reconstruction of an average cortical column in silico. *Brain Research Reviews*, 55:193–203, 2007.

Michael J. Higley and Diego Contreras. Integration of Synaptic Responses to Neighboring Whiskers in Rat Barrel Cortex In Vivo. *Journal of Neurophysiology*, 93:1920–34, 2005.

M. L. Hines and N. T. Carnevale. NEURON: a Tool for Neuroscientists. *Neuroscientist*, 7:123–135, 2001.

M. L. Hines and N. T. Carnevale. Translating network models to parallel hardware in NEURON. *Journal of Neuroscience Methods*, 169:425–455, 2007.

Michael L. Hines and Nicholas T. Carnevale. *The NEURON Book*. Cambridge University Press, 2006.

Michael L. Hines, Thomas Morse, Michele Migliore, Nicholas T. Carnevale, and Gordon M. Shepherd. ModelDB: A Database to Support Computational Neuroscience. *Journal of Computational Neuroscience*, 17:7–11, 2004.

Tony Hoare and Robin Milner. Grand Challenges for Computing Research. *The Computer Journal*, 48:49–52, 2005.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117: 500–544, 1952.

Urs Hölzle and Luiz André Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2009.

Jonathan C. Horton and Daniel L. Adams. The cortical column: a structure without a function. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360: 837–862, 2005.

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, Johannes Schemmel, Gert Cauwenberghs, John Arthur, Kai Hynna, Fopefolu Folowosele, Sylvain Saighi, Teresa Serrano-Gotarredona, Jayawan Wijekoon, Yingxue Wang, and Kwabena Boahen. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5:1–21, 2011.

Eugene M. Izhikevich. Simple Model of Spiking Neurons. *Neural Networks, IEEE Transactions on*, 14:1569–1572, 2003.

Eugene M. Izhikevich. Which Model to Use for Cortical Spiking Neurons? *Neural Networks, IEEE Transactions on*, 15:1063–1070, 2004.

Eugene M. Izhikevich. *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. The MIT Press, 2007.

Eugene M. Izhikevich and Gerald M. Edelman. Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences*, 105:3593–3598, 2008.

Xin Jin, Steve B. Furber, and John V. Woods. Efficient Modelling of Spiking Neural Networks on a Scalable Chip Multiprocessor. In *Neural Networks, International Joint Conference on*, pages 2812–2819, 2008.

Xin Jin, Francesco Galluppi, Cameron Patterson, Alexander Rast, Sergio Davies, Steve Temple, and Steve Furber. Algorithm and Software for Simulation of Spiking Neural Networks on the Multi-Chip SpiNNaker System. In *Neural Networks, International Joint Conference on*, pages 1–8, 2010.

Christopher Johansson and Anders Lansner. Towards cortex sized artificial neural systems. *Neural Networks*, 20:48–61, 2007.

Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell. *Principles of Neural Science*. McGraw-Hill Medical, 2000.

Dennis Kätzel, Boris V. Zemelman, Christina Buetfering, Markus Wölfel, and Gero Miesenböck. The columnar and laminar organization of inhibitory connections to neocortical excitatory cells. *Nature Neuroscience*, 14:100–107, 2011.

M. M. Khan, A. D. Rast, J. Navaridas, X. Jin, M. Luján, S. Temple, C. Patterson, D. Richards, J. V. Woods, J. Miguel-Alonso, and S. B. Furber. Event-driven configuration of a neural network CMP system over an homogeneous interconnect fabric. *Parallel Computing*, 97:392–409, 2009.

Harold T. Kyriazi and Daniel J. Simons. Thalamocortical Response Transformations in Simulated Whisker Barrels. *The Journal of Neuroscience*, 13:1601–1615, 1993.

Stefan Lang, Vincent J. Dercksen, Bert Sakmann, and Marcel Oberlaender. Simulation of signal flow in 3D reconstructions of an anatomically realistic neural network in rat vibrissal cortex. *Neural Networks*, 24:998–1011, 2011.

Victor W. Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthony D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal, and Pradeep Dubey. Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU. In *Computer Architecture, Internation Symposium on*, pages 451–460, 2010.

Sandrine Lefort, Christian Tomm, J.-C. Floyd Sarria, and Carl C. H. Petersen. The Excitatory Neuronal Network of the C2 Barrel Column in Mouse Primary Somatosensory Cortex. *Neuron*, 61:301–316, 2009.

Wolfgang Maass and Henry Markram. Theory of the Computational Function of Microcircuit Dynamics. In *Microcircuits: The Interface between Neurons and Global Brain Function*, chapter 18, pages 371–392. The MIT Press, 2006.

Henry Markram. The Blue Brain Project. *Nature Reviews Neuroscience*, 7:153–160, 2006.

Carver Mead. *Analog VLSI and neural systems*. Addison-Wesley, 1989.

Hanno S. Meyer, Verena C. Wimmer, Mike Hemberger, Randy M. Bruno, Christiaan P.J. de Kock, Andreas Frick, Bert Sakmann, and Moritz Helmstaedter. Cell Type-Specific Thalamic Innervation in a Column of Rat Vibrissal Cortex. *Cerebral Cortex*, 20:2287–2303, 2010a.

Hanno S. Meyer, Verena C. Wimmer, M. Oberlaender, Christiaan P.J. de Kock, Bert Sakmann, and Moritz Helmstaedter. Number and Laminar Distribution of Neurons in a Thalamocortical Projection Column of Rat Vibrissal Cortex. *Cerebral Cortex*, 20: 2277–2286, 2010b.

Vernon B. Mountcastle. Modality and topographic properties of single neurons of cat's somatic sensory cortex. *Journal of Neurophysiology*, 20:408–434, 1956.

Jayram Moorkanikara M. Nageswaran, Nikil Dutt, Jeffrey L. Krichmar, Alex Nicolau, and Alexander V. Veidenbaum. A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Networks*, 22:791–800, 2009.

Javier Navaridas, Mikel Luján, Jose Miguel-Alonso, Luis A. Plana, and Steve Furber. Understanding the Interconnection Network of SpiNNaker. In *Supercomputing, International Conference on*, pages 286–295, 2009.

Javier Navaridas, Luis A. Plana, Jose Miguel-Alonso, Mikel Luján, and Steve B. Furber. SpiNNaker: impact of traffic locality, causality and burstiness on the performance of the interconnection network. In *Computing Frontiers, International Conference on*, pages 11–20, 2010.

Andrew Nere, Sean Franey, Atif Hashmi, and Mikko Lipasti. Simulating cortical networks on heterogeneous multi-GPU systems. *Journal of Parallel and Distributed Computing*, 2012. In press.

Samuel A. Neymotin, Lee Heekyung, Eunhye Park, André A. Fenton, and William W. Lytton. Emergence of physiological oscillation frequencies in a computer model of neocortex. *Frontiers in Computational Neuroscience*, 5:1–17, 2011.

John Nolte. *Elsevier's Integrated Neuroscience*. Elsevier, 2007.

Nvidia. Online technical specification, 2013. URL `http://www.nvidia.co.uk/object/tesla_c1060_uk.html`. Retrieved on the 23rd of May 2013.

Marcel Oberlaender, Christiaan P. J. de Kock, Randy M. Bruno, Alejandro Ramirez, Hanno S. Meyer, Vincent J. Dercksen, Moritz Helmstaedter, and Bert Sakmann. Cell Type-Specific Three-Dimensional Structure of Thalamocortical Circuits in a Column of Rat Vibrissal Cortex. *Cerebral Cortex*, 22:2375–2391, 2011.

Adam M. Packer and Rafael Yuste. Dense, Unspecific Connectivity of Neocortical Parvalbumin-Positive Interneurons: A Canonical Microcircuit for Inhibition? *The Journal of Neuroscience*, 31:13260–13271, 2011.

Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Simon Davidson, Jeffrey Pepper, David Clark, Cameron Patterson, and Steve Furber. SpiNNaker: A Multi-Core System-on-Chip for Massively-Parallel Neural Net Simulation. In *Custom Integrated Circuits, Conference on*, pages 1–4, 2012.

Vivek K. Pallipuram, Mohammad A. Bhuiyan, and Melissa C. Smith. Evaluation of GPU Architectures using Spiking Neural Networks. In *Application Accelerators in High-Performance Computing, Symposium on*, pages 93–102, 2011.

Cameron Patterson, Jim Garside, Eustace Painkras, Steve Temple, Luis A. Plana, Javier Navaridas, Thomas Sharp, and Steve Furber. Scalable communications for a million-core neural processing architecture. *Journal of Parallel and Distributed Computing*, 72:1507–1520, 2012.

Martin Pearson, Ian Gilhespy, Kevin Gurney, Chris Melhuish, Benjamin Mitchinson, Mokhtar Nibouche, and Anthony Pipe. A Real-Time, FPGA Based, Biologically Plausible Neural Network Processor. In *Artificial Neural Networks, Internation Conference on*, pages 1021–1026, 2005.

Rodrigo Perin, Thomas K. Berger, and Henry Markram. A synaptic organizing principle for cortical neuronal groups. *Proceedings of the National Academy of Sciences*, 108: 5419–5424, 2011.

Alan Peters and Martin L. Feldman. The projection of the lateral geniculate nucleus to area 17 of the rat cerebral cortex. I. General description. *Journal of Neurocytology*, 5: 63–84, 1976.

Carl C. H. Petersen. The Functional Organisation of the Barrel Cortex. *Neuron*, 56: 339–353, 2007.

Elena Phoka, Mark Wildie, Simon R. Schultz, and Mauricio Barahona. Sensory experience modifies spontaneous state dynamics in a large-scale barrel cortical model. *Journal of Computational Neuroscience*, 33:323–339, 2012.

Filip Piękniewski. Online image, 2013. URL `http://www-users.mat.umk.pl/~philip/neuron.svg`. Retrieved on the 16th of May 2013.

David J. Pinto, Joshua C. Brumberg, Daniel J. Simons, and G. Bard Ermentrout. A Quantitative Population Model of Whisker Barrels: Re-Examining the Wilson-Cowan Equations. *Journal of Computational Neuroscience*, 3:247–264, 1996.

David J. Pinto, Joshua C. Brumberg, and Daniel J. Simons. Circuit Dynamics and Coding Strategies in Rodent Somatosensory Cortex. *Journal of Neurophysiology*, 83: 1158–1166, 2000.

David J. Pinto, Jed A. Hartings, Joshua C. Brumberg, and Daniel J. Simons. Cortical Damping: Analysis of Thalamocortical Response Transformations in Rodent Barrel Cortex. *Cerebral Cortex*, 13:33–44, 2003.

Luis A. Plana, Steve B. Furber, Steve Temple, Mukaram Khan, Yebin Shi, Jian Wu, and Shufan Yang. A GALS Infrastructure for a Massively Parallel Multiprocessor. *IEEE Design & Test of Computers*, 24:454–463, 2007.

Hans E. Plesser, Jochen M. Eppler, Abigail Morrison, Markus Diesmann, and Marc-Oliver Gewaltig. Efficient Parallel Simulation of Large-Scale Neuronal Networks on Clusters of Multiprocessor Computers. In *International Euro-Par Conference*, 2007.

Kenneth L. Rice, Mohammad A. Bhuiyan, Tarek M. Taha, Christopher N. Vutsinas, and Melissa C. Smith. FPGA Implementation of Izhikevich Spiking Neural Networks for Character Recognition. In *Reconfigurable Computing and FPGAs, International Conference on*, pages 451–456, 2009.

Alex Roxin. The role of degree distribution in shaping the dynamics in networks of sparsely connected spiking neurons. *Frontiers in Computational Neuroscience*, 5: 1–15, 2011.

Valentina Salapura, Randy Bickford, Matthias Blumrich, Arthur A. Bright, Dong Chen, Paul Coteus, Alan Gara, Mark Giampapa, Michael Gschwind, Manish Gupta, Shawn Hall, Ruud A. Haring, Philip Heidelberger, Dirk Hoenicke, Gerard V. Kopcsay, Martin Ohmacht, Rick A. Rand, Todd Takken, and Pavlos Vranas. Power and Performance Optimization at the System Level. In *Computing Frontiers, Internation Conference on*, pages 125–132, 2005.

Johannes Schemmel, Johannes Fieres, and Karlheinz Meier. Wafer-Scale Integration of Analog Neural Networks. In *Neural Networks, International Joint Conference On*, pages 431–438, 2008.

Johannes Schemmel, Daniel Briiderle, Andreas Griibl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neural Modeling. In *Circuits and Systems, International Symposium on*, pages 1947–1950, 2010.

Thomas Sharp and Steve Furber. Correctness and Performance of the SpiNNaker Architecture. In *Neural Networks, International Joint Conference on*, 2013.

Thomas Sharp, Cameron Patterson, and Steve Furber. Distributed Configuration of Massively-Parallel Simulation on SpiNNaker Neuromorphic Hardware. In *Neural Networks, International Joint Conference on*, 2011a.

Thomas Sharp, Luis A. Plana, Francesco Galluppi, and Steve Furber. Event-Driven Simulation of Arbitrary Spiking Neural Networks on SpiNNaker. In *Neural Information Processing, International Conference on*, pages 424–430, 2011b.

Thomas Sharp, Francesco Galluppi, Alexander Rast, and Steve Furber. Power-efficient simulation of detailed cortical microcircuits on SpiNNaker. *Journal of Neuroscience Methods*, 210:110–118, 2012.

Daniel J. Simons and George E. Carvell. Thalamocortical Response Transformation in the Rat Vibrissa/Barrel System. *Journal of Neurophysiology*, 61:311–330, 1989.

Josef Skrzypek, editor. *Neural Network Simulation Environments*. Springer, 1994.

William R. Softky and Christof Koch. The Highly Irregular Firing of Cortical Cells Is Inconsistent Temporal Integration of Random EPSPs. *The Journal of Neuroscience*, 13:334–350, 1993.

Armen Stepanyants, Judith A. Hirsch, Luis M. Martinez, Zoltán F. Kisvárday, Alex S. Ferecskó, and Dmitri B. Chklovskii. Local potential connectivity in cat primary visual cortex. *Cerebral Cortex*, 18:13–28, 2008.

Thomas Sterling, Donald J. Becker, Daniel Savarese, John E. Dorband, Udaya A. Ranawake, and Charles V. Packer. BEOWULF: A Parallel Workstation For Scientific Computation. In *Parallel Processing, International Conference on*, pages 11–14, 1995.

Evangelos Stromatias, Francesco Galluppi, Cameron Patterson, and Steve Furber. Power analysis of large-scale, real-time neural networks on SpiNNaker. In *Neural Networks, International Joint Conference on*, 2013.

Qian-Quan Sun, John R. Huguenard, and David A. Prince. Barrel Cortex Microcircuits: Thalamocortical Feedforward Inhibition in Spiny Stellate Cells Is Mediated by a Small Number of Fast-Spiking Interneurons. *The Journal of Neuroscience*, 26: 1219–1230, 2006.

Andrew Symes and Thomas Wennekers. Spatiotemporal dynamics in the cortical microcircuit: A modelling study of primary visual cortex layer 2/3. *Neural Networks*, 22:1079–1092, 2009.

Alex M. Thomson. Interlaminar Connections in the Neocortex. *Cerebral Cortex*, 13: 5–14, 2003.

Alex M. Thomson and Christophe Lamy. Functional maps of neocortical local circuitry. *Frontiers in Neuroscience*, 1:19–42, 2007.

Alex M. Thomson, David C. West, Yun Wang, and A. Peter Bannister. Synaptic Connections and Small Circuits Involving Excitatory and Inhibitory Neurons in Layers 2–5 of Adult Rat and Cat Neocortex: Triple Intracellular Recordings and Biocytin Labelling In Vitro. *Cerebral Cortex*, 12:936–953, 2002.

Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.

Simon J. Thorpe, Arnaud Delorme, and Rufin VanRullen. Spike-based strategies for rapid processing. *Neural Networks*, 14:715–726, 2001.

Christian Tomm. *Analysing Neuronal Network Architectures: From Weight Distributions to Structure and Back*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2012.

Alan Turing. Can Digital Computers Think? Online radio-broadcast transcription, 1951. URL `http://www.turingarchive.org/browse.php/B/5`. Retrieved on the 16th of May 2013.

Nobuhiko Wagatsuma, Tobias C. Potjans, Markus Diesmann, and Tomoki Fukai. Layer-dependent attentional processing by top-down signals in a visual cortical microcircuit model. *Frontiers in Computational Neuroscience*, 5:1–15, 2011.

Verena C. Wimmer, Randy M. Bruno, Christiaan P. J. de Kock, Thomas Kuner, and Bert Sakmann. Dimensions of a Projection Column and Architecture of VPM and POm Axons in Rat Vibrissal Cortex. *Cerebral Cortex*, 20:2265–2276, 2010.

Thomas A. Woolsey and Hendrik Van der Loos. The Structural Organization of Layer IV In the Somatosensory Region (S I) of Mouse Cerebral Cortex. *Brain Research*, 17: 205–242, 1970.

Pierre Yger, Sami El Boustani, Alain Destexhe, and Yves Frégnac. Topologically invariant macroscopic statistics in balanced networks of conductance-based integrate-and-fire neurons. *Journal of Computational Neuroscience*, 31:229–245, 2011.