High-Performance Computing for Systems of Spiking Neurons

Steve Furber and Steve Temple[*]
[*]The University of Manchester
School of Computer Science
Oxford Road, Manchester M13 9PL, UK
steve.furber,steven.temple@manchester.ac.uk

Andrew Brown[†]
[†]The University of Southampton
Department of Electronics and Computer Science
Southampton, Hampshire, SO17 1BJ, UK
adb@ecs.soton.ac.uk

Abstract

We propose a bottom-up computer engineering approach to the Grand Challenge of understanding the Architecture of Brain and Mind as a viable complement to top-down modelling and alternative approaches informed by the skills and philosophies of other disciplines. Our approach starts from the observation that brains are built from spiking neurons and then progresses by looking for a systematic way to deploy spiking neurons as components from which useful information processing functions can be constructed, at all stages being informed (but not constrained) by the neural structures and microarchitectures observed by neuroscientists as playing a role in biological systems. In order to explore the behaviours of large-scale complex systems of spiking neuron components we require high-performance computing equipment, and we propose the construction of a machine specifically for this task – a massively parallel computer designed to be a universal spiking neural network simulation engine.

1 Introduction

1.1 Neurons

The basic biological control component is the neuron. A full understanding of the 'Architecture of Brain and Mind' (Sloman, 2004) must, ultimately, involve finding an explanation of the phenomenological observations that can be expressed in terms of the interactions between the neurons that comprise the brain (together with their sensory inputs, actuator outputs, and related biological processes).

Neurons appear to be very flexible components whose utility scales over systems covering a vast range of complexities. Very simple creatures find a small number of neurons useful, honey bees find it economic to support brains comprising around 850,000 neurons, and humans have evolved to carry brains comprising 10^{11} neurons or so. The component neuron used this range of complexities is basically the same in its principles of operation, so in some sense it has a universality similar to that enjoyed by the basic logic gate in digital engineering.

There is a further similarity between neurons and logic gates: both are multiple-input single-output components. However, while the typical fan-in (the number of inputs to a component) and fan-out (the number of other components the output of a particular component connects to) of a logic gate is in the range 2 to 4, neurons typically have a fan-in and fan-out in the range 1,000 to 10,000. (It is easy to show that that mean fan-in and fan-out in a system are the same – they are just different ways of counting the number of connections between components.)

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

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

1.2 Neural microarchitecture

The universality of the neuron as a component is also reflected in certain higher-level structures of the brain. For example, the cortex displays a 6-layer structure and a regularity of interconnect between the neurons in the various layers that can reasonably deserve the application of the term 'microarchitecture'. The same regular laminar cortical microarchitecture is in evidence across the cortex in regions implementing low-level vision processes such as edge-detection and in regions involved in high-level functions such as speech and language processing. This apparent 'universality' (used here to describe one structure that can perform any function) of the cortical microarchitecture suggests there are principles being applied here the understanding of which could offer a breakthrough in our understanding of brain function.

In contrast to the regularity and uniformity of the microarchitecture, the particular connectivity patterns that underpin these structures appear to be random, guided by statistical principles rather than specific connectivity plans. The connectivity is also locally adaptive, so the system can be refined through tuning to improve its performance.

1.3 Engineering with neurons

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

There are other 'engineering' aspects of biological neurons that are interesting, too. We have already mentioned the regularity of neural microarchitecture. The power-efficiency of neurons (measured as the energy required to perform a given computation) exceeds that of computer technology, possibly because the neuron itself is a very low performance component. While computer engineers measure gate speeds in picoseconds, neurons have time constants of a millisecond or longer. While computer engineers worry about speed-of-light limitations and the number of clock cycles it takes to get a signal across a chip, neurons communicate at a few metres per second. This very relaxed performance at the technology level is, of course, compensated by the very high levels of parallelism and connectivity of the biological system. Finally, neural systems display levels of fault-tolerance and adaptive learning that artificial systems have yet to approach.

We have therefore decided to take up the challenge to find ways to build useful systems based upon spiking neuron components (for example, Furber, Bainbridge, Cumpstey and Temple, 2004), and we hope that this will lead to mutually-stimulating interactions with people from many other disciplines whose approach to the same Grand Challenge, of understanding the Architecture of Brain and Mind, will be quite different from our own.

2 Relevance to GC5

What has any of this engineering really got to do with the Grand Challenge of understanding the Architecture of Brain and Mind?

As this is aimed at a broad audience, not many of whom are computer engineers, we will digress briefly to consider what computer engineers may bring to this Grand Challenge. To begin with, it is useful to appreciate the skills and mindset that a computer engineer, for better or for worse, possesses. What can a person whose stock-in-trade consists of logic gates, microchips and printed circuit boards contribute to the bio-psycho-philosophical quest to understand the workings of the mind?

2.1 A Computer Engineer's manifesto

To a computer engineer 'understand' has a specific meaning that is different from what a scientist means by the same word, which is in turn probably different from the meanings used by other disciplines. To the scientist, understanding is to have a repeatably-verifiable explanation of a phenomenon. To the engineer, understanding means to be able to go away and build another artefact that works in the same way. The scientist's analysis reduces a complex phenomenon into its basic components; this is complemented by the engineer's ability to take those components, or components that encapsulate the same essential behaviour, and synthesize them back into a functioning system.

Thus, when a computer engineer claims to 'understands' how a mobile phone works, the statement can be interpreted as meaning that they can (at least in principle) explain when every one of the 100 million or so transistors switches, why it switches, what will happen if it fails to switch, and so on. OK, we might get on less secure ground when describing the chemistry of the lithium-ion battery and the details of the radio and antenna design or the higher levels of the software. And when it comes to explaining why the plastic case is pink and the buttons are arranged in swirling patterns with no obvious ergonomic objective we are completely lost! But back in the familiar territory of the digital transistor circuits we have a vocabulary comprising baseband processors, DSPs, maximum likelihood error correctors, RAMs, buses, interrupts, and so on, that together provide a language of description at multiple levels of abstraction from an individual transistor to the lower levels of the system software. This enables us to describe in very fine detail how the phone works and, more particularly, how you might make another working phone at lower cost and with better battery life.

This is the approach we bring to understanding the Architecture of Brain and Mind. In neuroscience we see that there are pretty accurate models of the basic component from which brains are built – the neuron. There are some rather sketchy and limited descriptions of how these components are interconnected and how they behave in natural networks, and there is rather better information about their macro-level modular organisation and gross activity. The weakest part of the neuroscientists' analysis (for very good reason – it is hard to apply reductionist principles to systems whose interesting characteristics depend on their organizational complexity) is at the intermediate levels between the component neurons (where analysis is applicable) and the macro-organisation (where mean field statistics work).

This intermediate level is precisely the level at which the computer engineer may have something to offer. Assembling basic components into functional units, implementing useful computational processes based on networks of dynamical systems, these are all second nature to the computer engineer once we have come to grips with the spiking neuron as a component. As we observed earlier, it even looks a bit like a logic gate – several inputs but only one output.

The intrinsic dynamics of a neuron may confound the computer engineer who is used to working only with digital circuits that are controlled by the extrinsic straitjacket of a clock signal, but a small minority of us are proficient in building circuits whose sequential behaviour is intrinsic – members of the class of digital circuit generally described as asynchronous or self-timed. The knowledge we hold on how to build reliable, highly complex asynchronous digital systems *may just* provide us with new insights into the highly complex asynchronous neural systems that provide the hardware platform upon which the brain and mind are built.

2.2 GC5 methodology

Our approach to this Grand Challenge is essentially bottom-up, which will complement the top-down and middle-out approaches that are better-suited to those who bring different skills and mindsets from other disciplines.

The bottom-up approach starts from the concept of a neuron as a basic component, and then seeks useful compositions of neurons to create (and implement) increasingly higher levels of functional abstraction. These compositions may be inspired by neuroscience; for example, we have an involvement in the EPSRC-funded COLAMN project which has as its goal the creation of novel computational architectures based on the laminar microarchitecture of the neocortex, with considerable input from the 'wet' neuroscientists in the project. Or they may be designed in the abstract; for example our earlier work on *N*-of-*M* coded sparse distributed memories (Furber, Bainbridge, Cumpstey and Temple, 2004) – with at best tenuous relevance to biology.

A feature of this research is that it can yield a positive outcome in two distinct ways. It may contribute to the scientific objective of understanding the architecture of brain and mind, and/or it may contribute to the engineering objective of delivering better/different/novel models of computation. Either of these outcomes would justify our engagement, and with a following wind we might just achieve both...

In order to pursue this research agenda we need a sandpit in which we can experiment with neuron components on a large scale, hence the massively parallel high-performance computer theme that we will turn to shortly. This large-scale engineering project brings with it additional research aspects relating to fault-tolerance, autonomic computing, self-healing, networks-on-chip, and so forth, all of which add to the engineering challenge but probably contribute little to the GC5 agenda.

3 Objectives

We have set ourselves the objective of simulating a billion spiking neurons in real time while making as few assumptions as possible about what a neuron is and how the neurons are connected. We approach this by viewing a neural system as an event-driven dynamical system – a hybrid system where a (large) set of components, each of which operates in continuous time (and is characteristically described by a set of differential equations), interact through discrete events.

In order to retain complete flexibility in the internal neural dynamics we implement the real-time differential equation solvers (which will typically use discrete-time fixed-point approximations) in software, and then exploit the high speeds of electronic signalling to communicate the discrete inter-neuron communication events around the system in a time which is close to instantaneous on the time-scales of the neuron dynamics. This allows us to use a virtual mapping from the physical structure of the biological system we are modelling to the physical structure of the electronic system we are running the model on.

4 Neural computation

Any computation system must achieve a balance between its processing, storage and communication functions. It is useful to consider how these three functions are achieved in neural systems.

4.1 Processing

The neuron itself performs the processing function. It produces output events in response to input events through a non-linear transfer function, which we will model using suitable differential equations whose complexity is limited only by the available computing power.

The simplest neuron models process inputs by taking a linear sum of the inputs, each weighted by the strength of its respective synapse. When the inputs are spike events the multiplication implied by the weighting process reduces to repeated addition. Multiplication by repeated addition is usually inefficient, but here many inputs are likely to be inactive at any time and multiplication by zero by repeated addition is supremely efficient!

The weighted input sum is then used to drive the neural dynamics. A leaky-integrate-and-fire (LIF) model applies an exponential decay to the effect of each input, but if enough inputs fire close together in time to push the total activation past a threshold, the neuron fires its output and resets. More sophisticated models have more complex dynamics. For example, the models by Izhikevich (2004) are based on mathematical bifurcation and display a more diverse range of biologically-relevant behaviours that the LIF model.

4.2 Communication

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

Since the spike carries no information in its shape or size, the only information is which neuron fired and when it fired. In a real-time simulation the timing is implicit (and the communication, being effectively instantaneous, preserves the timing), so all we need to communicate is the identity of the neuron that fired, and we must send that to every neuron to which the firing neuron connects.

In the biological system the identity of a firing neuron is spatially encoded – each neuron has its own physical axon. In our system we cannot implement an equivalent level of physical connectivity so instead we use logical encoding by sending a packet identifying the firing neuron around a network that connects all of the components together.

4.3 Storage

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

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

The primary long-term storage mechanism is synaptic modification (within which we include the growth of new synapses).

In a real-time modelling system we expect the modelling to capture the neural and networks dynamics, and hence the contributions these mechnisms make to information storage. The axon delay-line storage does not come so easily as we have deliberately exploited the high speeds of electronic signalling to make spike communication effectively instantaneous in order to support a virtual mapping of the physical structures. It is likely that the axon delay is functionally important, so we must put these delays back in, either by delaying the issue of the spike or by delaying its effect at the destination. Either solution can be achieved in software, but both have drawbacks, and this remains one of the trickier aspects of the design to resolve to our complete satisfaction.

The final storage process is the most fundamental: synaptic weight adaptivity. Here we require long-term stability and support for a range of learning algorithms. We will exploit the fact that digital semiconductor memory is a mass-produced low-cost commodity, and the proposed machine is built around the use of commodity memory for storing synaptic connectivity information.

Indeed, as we shall see in the next section, the major resources in a neural computation system revolve around the synapses, not around the neural dynamics.

5 Computing requirements

Various estimates have been offered for the computational power required to run a real-time simulation of the human brain based on reasonably realistic neuron models. The answer generally comes out in the region of 10^{16} instructions per second, which is some way beyond the performance of a desktop PC or workstation, but not far beyond the performance of the petaFLOP supercomputers currently in design.

The route to this performance estimate can be summarized as follows: the brain comprises around 10^{11} neurons, each with of the order of 1,000 inputs. Each input fires at an average rate of 10 Hz, giving 10^{15} connections per second, and each connection requires perhaps 10 instructions.

Note that this estimate is based on the computing power required to handle the synaptic connections. Modelling the neuron dynamics is a smaller part of the problem: 10^{11} neurons each requiring a few 10s of instructions to update their dynamics perhaps 10^3 times a second, requiring in total an order of magnitude less computing power than the connections.

A similar calculation yields the memory requirements of such a simulation: 10^{14} synapses each require of the order of a few bytes, so around 10^{14} bytes of synaptic connection data are required.

At present the only way a machine of such capacity can be conceived is to employ a massively parallel architecture. This is likely to remain true even with future developments in CMOS technology as further increases in clock speed and individual processor throughput are unlikely to be great, as evidenced by the recent trend towards multi-core processors from all of the leading microprocessor vendors. The future of the microprocessor is in chip multiprocessors, and the future of high-performance computing is in massively parallel systems.

Fortunately, the problem of simulating very large numbers of neurons in real time falls into the class of 'embarrassingly' parallel applications, where the available concurrency allows the trade-off of processor performance against the number of processors to be totally flexible. The issue, then, is to determine how such a system might be optimised. What are the relevant metrics against which to make decisions on the systems architecture?

We propose that the primary metrics should be *performance density* (measured in MIPS/mm^2 of silicon) and *power-efficiency* (measured in MIPS/watt). The former is the primary determinant of the capital cost of the machine, while the latter influences both the capital cost – in terms of the cooling plant – and the running cost – a machine such as this demands a significant electrical power budget.

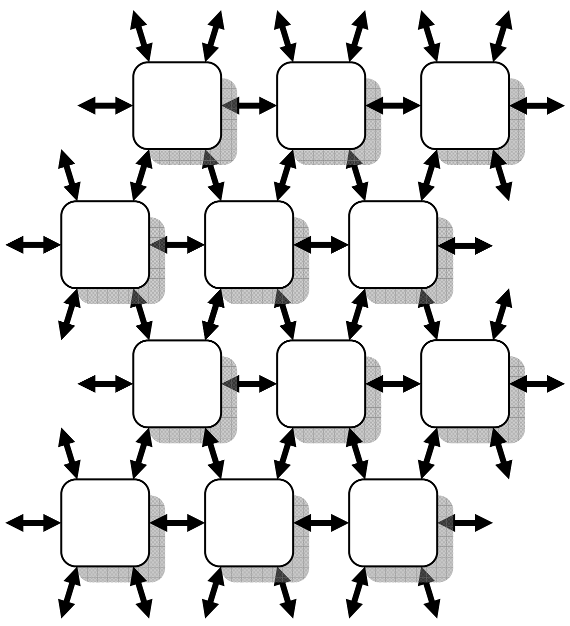
Figure 1: The system architecture.

A choice then has to be made between using a large number of high-performance processors or an even larger number of more power-efficient embedded processors. Here the metrics can be our guide – embedded processors win handsomely on power-efficiency, and to a lesser extent on performance density, over their much more complex high-end counterparts.

That, then sets the course for this work. The objective is to build a machine, based on large numbers of small processors, that has the potential to scale up to the levels of parallelism and performance necessary to model a brain in real time. Admittedly, modelling a complete human brain is some way beyond our current goals, but we should be able to model substantial parts of the human brain and complete brains of less complex species with what we propose here, which is a machine capable of modelling a billion spiking neurons in real time.

6 SpiNNaker

A spinnaker is a large foresail that enables a yacht to make rapid progress in a following wind (see reference to 'following wind' in Section 2.2 above!). We have adopted SpiNNaker as a name for our project because it comes close to a contraction of 'a (uni-

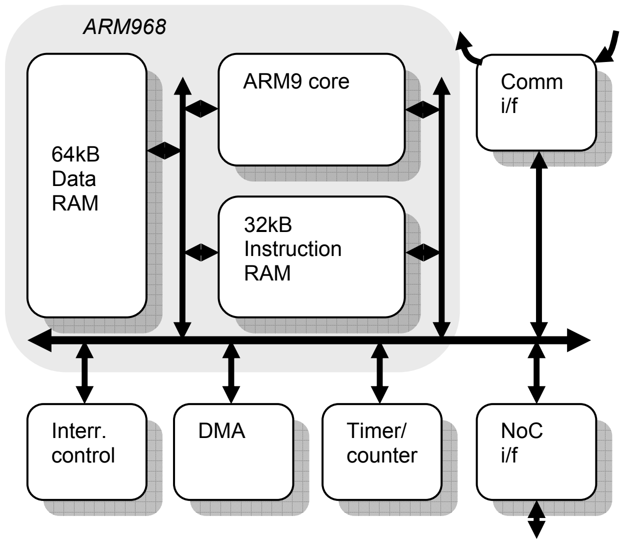

Figure 2: Processor subsystem organization.

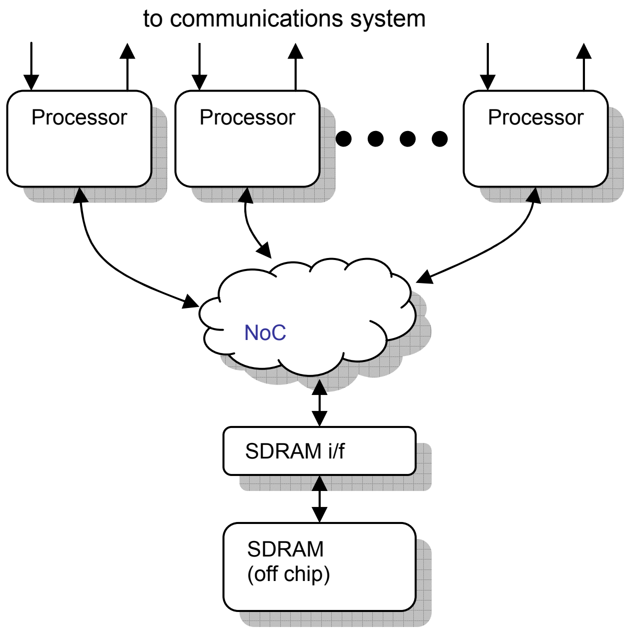

Figure 3: The CMP system NoC.

versal) Spiking Neural Network architecture', provided you say it quickly enough. Again, this is our goal: to build a computer system that is as universal as we can make it in its ability to simulate large systems of spiking neurons, preferably in real time.

The following description of the system is largely extracted from Furber, Temple and Brown (2006).

6.1 System architecture

The system is implemented as a regular 2D array of nodes interconnected through bi-directional links in a triangular formation as illustrated in Fig. 1. The 2D mesh is very straightforward to implement on a circuit board and also provides many alternative routes between any pair of nodes which is useful for reconfiguration to isolate faults. (Nothing in the communications architecture precludes the use of a more complex topology if this proves advantageous.)

Each node in the network comprises two chips: a chip multiprocessor (CMP) and an SDRAM, with the integer processing power of a typical PC but at much lower power and in a compact physical form. The six bidirectional links support a total of 6 Gbit/s of bandwidth into and out of the node. A system of 100 x 100 nodes will deliver a total of 40 teraIPS, sufficient to simulate perhaps 200 million spiking neurons in real time, and will have a bisection bandwidth of 200 Gbit/s.

6.2 ARM968 processor subsystem

For the reasons already outlined, we choose to base the system around a massively-parallel array of power-efficient embedded processors, and have chosen the ARM968 as offering the best balance of performance, area, power-efficiency and ease of use for our purposes. The ARM968 is a synthesizable ARM9 processor core with tightly-coupled instruction and data memories, and an integral on-chip bus (ARM Ltd, 2004). Each processor subsystem comprises a processor, instruction and data memory, timers, interrupt and DMA controllers and a communications NoC interface (Fig. 2).

We estimate that a 200 MIPS integer embedded ARM9 processor should be able to model 1,000 leaky-integrate-and-fire (or Izhikevich) neurons, each with 1,000 inputs firing on average at 10 Hz, in real time. The synaptic connectivity information for these neurons requires around 4 Mbytes of memory and the neuron state requires around 50 Kbytes of memory. These estimates have led us to adopt a hybrid architecture where the synaptic data is held in an off-chip SDRAM while the neural state data is held in on-chip memory local to each embedded processor. A processing node in our system therefore comprises two ICs: a chip multiprocessor (CMP) with about twenty 200 MIPS embedded ARM9 processors, and an SDRAM chip. The synaptic data is accessed in large blocks and this enables an SDRAM bandwidth of around 1 GByte/s to provide this data at the required rate.

The processors on a CMP share access to the SDRAM using a self-timed packet-switched Network-on-Chip (NoC). This fabric will use the CHAIN technology (Bainbridge and Furber, 2002), developed at the University of Manchester and commercialized by Silistix Ltd, which gives a through-

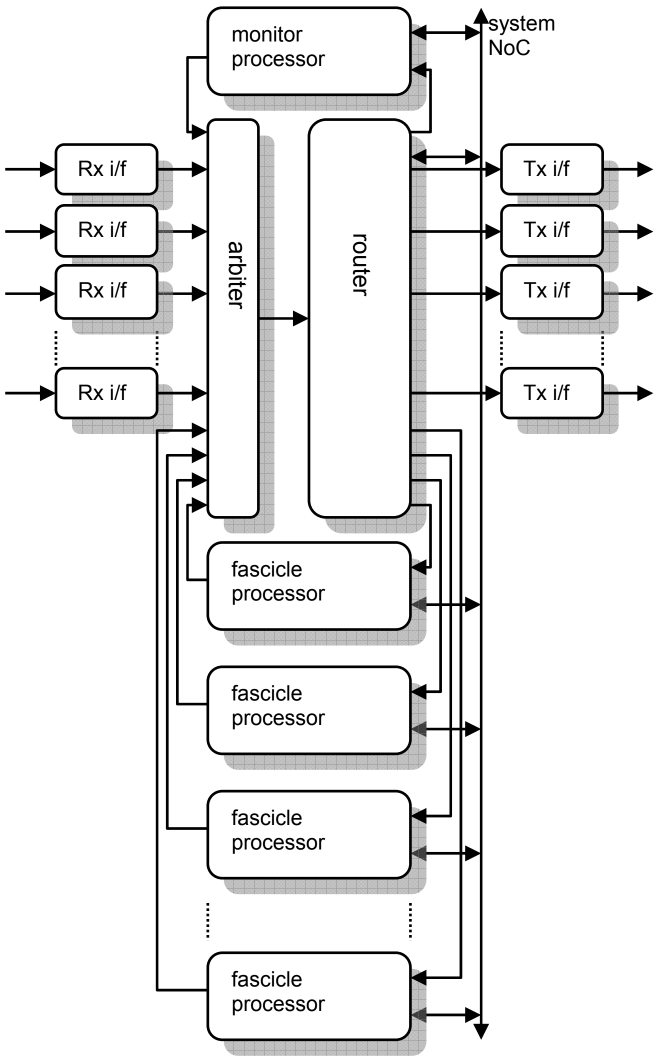

Figure 4: The communications NoC.

put of around 1 Gbit/s per 6-wire link (Bainbridge, Plana and Furber, 2004). The organization of the system NoC that connects the processor subsystems to the SDRAM is shown in Fig. 3.

6.3 The communications system

The major challenge in designing a scalable multi-chip neural modeling system is to emulate the very high connectivity of the biological system. The high fan-in and fan-out of neurons suggests that an efficient multicast communication system is required. We propose a communication NoC fabric based upon address-event signaling, but carried over a second self-timed packet-switched fabric rather than the usual bus-based fabric. The self-timed fabric decouples the many different clock domains within and across the CMPs.

The inter-chip communication uses a self-timed signalling system on an 8-wire inter-chip link that employs a self-timed 2-of-7 non-return-to-zero (NRZ) code (Bainbridge, Toms, Edwards and Furber, 2003) with an NRZ acknowledge. 16 of the 21 possible 2-of-7 codes are used to carry four bits of data, and a 17th code carries end-of-packet (EOP). Each 8-wire link has a capacity of around 1 Gbit/s when connecting two CMPs on the same circuit board, matching the on-chip bandwidth of a CHAIN link, and the self-timed protocol guarantees correct operation (albeit at a lower data rate) when the CMPs are on different circuit boards, automatically adapting to the addition delays incurred by any signal buffering that may be required.

The complete communications subsystem on a CMP is illustrated in Fig. 4. The inter-chip links are accessed via input protocol converters ('Rx i/f' in Fig. 4) that translate the off-chip 2-of-7 NRZ codes to the on-chip CHAIN codes, and output protocol converters ('Tx i/f') that perform the inverse translation. Each of the on-chip processing subsystems ('fascicle processor') is also a source of network traffic and a potential destination. All of the on- and off-chip sources are merged through an asynchronous arbiter into a single stream of packets that passes through the multicast router which will, in turn, propagate the packet to a subset of its on- and off-chip outputs. The monitor processor is identical to a fascicle processor but is dedicated to system management functions rather than neural modeling. It is chosen from among the fascicle processors at boot time; the flexibility in its selection removes another possible single point of failure on the CMP, improving fault tolerance.

The heart of the communication subsystem is the associative multicast router which directs every incoming packet to one or more of the local processors and output links using a routing key based on the source ID and a route look-up table.

6.4 Fault-tolerance

The scale of the proposed machine demands that it be designed with a high degree of fault-tolerance. Since the neural system we are modelling has intrinsic fault-tolerant properties (healthy humans lose about one neuron a second throughout their adult life; neurodegenerative diseases incur much higher loss rates) this capacity will be transferred to the simulated system to some degree. However, many of the techniques we employ to map the natural system onto the electronic model concentrate distributed biological processes into single points of failure in the model: a single microprocessor models a thousand neurons; a single inter-chip link carries the spikes on perhaps a million axons. Thus we must engineer some additional resilience into the electronic system.

The highly regular structure of the machine comes to our aid here. If a processor fails we can migrate its workload to another, on the same or on a different chip. This will almost certainly lead to a glitch in the system's real-time performance, but our goal is to minimise the size of this glitch and to build a system that is continuously monitoring its own performance and migrating its workload to minimise congestion, so a major failure just puts a higher transient demand on the workload management processes.

An inter-chip link failure (whether permanent or transient, perhaps due to local congestion) will be handled in the first instance at the hardware level, redirecting traffic automatically via an adjacent link, before invoking the performance management software to carry out a more permanent solution.

At all stages in the design we are exploring opportunities to identify mechanisms that support real-time fault-tolerance, some of which exploit the intrinsic fault-tolerance of neural systems but many of which will contribute to a separate research agenda in the area of autonomic, self-healing systems.

7 Conclusions

The Grand Challenge of understanding the Architecture of Brain and Mind is a multidisciplinary quest that will require many complementary approaches to run concurrently, each feeding off the others as sources of inspiration, ideas and sanity checks. The system synthesis approach of computer engineers such as ourselves may have something to contribute as a component of the overall process. An understanding of complex asynchronous interactions within digital systems seems highly relevant to the task of understanding the complex asynchronous interactions between neurons.

In our quest to understand the dynamics of systems of asynchronous spiking neurons we hope to contribute both to providing tools that help understand biological brains and also to the creation of novel computational systems that are inspired by biology, but whose link to biology may ultimately become tenuous.

To this end we propose to construct a massively-parallel computer that implements a universal spiking neural network architecture, SpiNNaker. Based on a chip multiprocessor incorporating around twenty 200 MIPS embedded ARM968 processors, and employing a communications infrastructure specifically designed to support the multicast routing required for neural simulation, this system will scale to hundreds of thousands of processors modelling up to a billion neurons in real time. It will form a 'sandpit' in which we, and others with similar interests, can experiment with large-scale systems of spiking neurons to test our network topologies and neural models in order to validate (or disprove) our theories of how neurons interact to generate the hardware platform that underpins the Architecture of Brain and Mind.

Acknowledgements

This work is supported by the EPSRC Advanced Processor Technologies Portfolio Partnership Award. Steve Furber holds a Royal Society-Wolfson Research Merit Award. The support of ARM Ltd and Silistix Ltd for the proposed work is gratefully acknowledged.

References

ARM Ltd. ARM968E-S Technical Reference Manual. DDI 0311C, 2004. http://www.arm.com/products/CPUs/ARM968E-S.html

W. J. Bainbridge and S. B. Furber. CHAIN: A Delay-Insensitive Chip Area Interconnect. *IEEE Micro*, 22(5):16-23, 2002.

W. J. Bainbridge, L. A. Plana and S. B. Furber. The Design and Test of a Smartcard Chip Using a CHAIN Self-timed Network-on-Chip. *Proc. DATE'04*, 3:274, Paris, Feb 2004.

W. J. Bainbridge, W. B. Toms, D. A. Edwards and S. B. Furber. Delay-Insensitive, Point-to-Point Interconnect using m-of-n codes. *Proc. Async* :132-140, Vancouver, May 2003.

S. B. Furber, W. J. Bainbridge, J. M. Cumpstey and S. Temple. A Sparse Distributed Memory based upon N-of-M Codes. *Neural Networks* 17(10):1437-1451, 2004.

S. B. Furber, S. Temple and A. D. Brown. On-chip and Inter-Chip Networks for Modelling Large-Scale Neural Systems. *Proc. ISCAS'06*, Kos, May 2006 (to appear).

E. M. Izhikevich. Which Model to Use for Cortical Spiking Neurons? *IEEE Trans. Neural Networks,* 15:1063-1070, 2004.

A. Sloman (ed.). The Architecture of Brain and Mind. UKCRC Grand Challenge 5 report, 2004.