ARTICLE IN PRESS

J. Parallel Distrib. Comput. ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Scalable communications for a million-core neural processing architecture

Cameron Patterson *, Jim Garside *, Eustace Painkras, Steve Temple, Luis A. Plana, Javier Navaridas, Thomas Sharp, Steve Furber

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

ARTICLE INFO

Article history:
Received 22 July 2011
Received in revised form
16 December 2011
Accepted 29 January 2012
Available online xxxx

Keywords:
GALS
HPC
Network-on-Chip
Neuromorphic
Parallel architecture
Low-power

ABSTRACT

The design of a new high-performance computing platform to model biological neural networks requires scalable, layered communications in both hardware and software. SpiNNaker's hardware is based upon Multi-Processor System-on-Chips (MPSoCs) with flexible, power-efficient, custom communication between processors and chips. The architecture scales from a single 18-processor chip to over 1 million processors and to simulations of billion-neuron, trillion-synapse models, with tens of trillions of neural spike-event packets conveyed each second. The communication networks and overlying protocols are key to the successful operation of the SpiNNaker architecture, designed together to maximise performance and minimise the power demands of the platform. SpiNNaker is a work in progress, having recently reached a major milestone with the delivery of the first MPSoCs. This paper presents the architectural justification, which is now supported by preliminary measured results of silicon performance, indicating that it is indeed scalable to a million-plus processor system.

1. Introduction

SpiNNaker (Fig. 1) is a novel application-specific architecture designed for simulation of massively-parallel *Spi*king *N*eural *N*etworks in real-time. Whilst it may be adapted to other computational purposes, the requirements of a neural computer have dominated its design. Biological neurons are slow, highly-interconnected units; electronic components are fast but have much lower fanout. The design therefore trades-off these properties using processors to simulate neuron and synaptic behaviour, and a fast network to deliver communications over packet-switched links.

Modelling 'brain-sized' networks in real-time requires a huge number of processors, with the network capacity to match. SpiNNaker is designed to be expandable to biologically-significant sizes [27] at reasonable cost, incorporating fault-tolerance and energy-efficiency as key aspects of the system design.

1.1. Network requirement

As far as is understood, biological neurons communicate primarily using 'spikes'; each spike is a 'digital' signal in that it is either present or not. Output variations are represented by the *temporal* spacing of spikes as discussed by Izhikevich [29]. Other connectivity information is in the *synaptic weight,* which indicates how strongly a spike affects each efferent neuron.

SpiNNaker represents a spike as a single, short communications packet, whose timing is represented by real time. A spike is *multicast* into the communications network where it may be replicated to a preprogrammed – and possibly large – set of destinations. (Typically in a biological neural network the input and output connection 'fan' of a neuron may be of the order 1000–10,000 and sometimes up to 250,000 [41].) This electronic transmission is nearly 'instantaneous' on biological timescales [11] and the 'real' biological delays are modelled by the receiving processors in software [49,32].

Depending on the neuron model used [32,48,47], a million-plus processor SpiNNaker system supports around 1 billion neurons in real-time, thus connectivity may exceed 1 trillion synapses. At an expected biological firing rate of 10 Hz [16] there could be 10 billion-plus neuron firings per second which amplify in the output fans to trillions of communication events/s. The network fabric has been scaled assuming significant locality of spike traffic (destination neurons are statistically proximate to the transmitting neuron), as is typically seen in the brain [8]. With these factors it is anticipated that the inter-chip links will not be a limiting factor in machine scalability. Thus, the connectivity problem is to distribute huge numbers of short packets very widely amongst up to a million processors efficiently, and in a timely fashion.

* Corresponding authors.
E-mail addresses: cameron.patterson@cs.man.ac.uk (C. Patterson), jdg@cs.man.ac.uk (J. Garside), painkras@cs.man.ac.uk (E. Painkras), temples@cs.man.ac.uk (S. Temple), plana@cs.man.ac.uk (L.A. Plana), javier.navaridas@cs.man.ac.uk (J. Navaridas), thomas.sharp@cs.man.ac.uk (T. Sharp), steve.furber@manchester.ac.uk (S. Furber).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.01.016

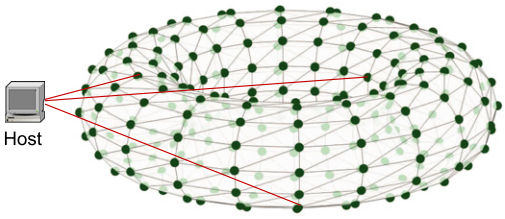


Fig. 1. Typical SpiNNaker system inter-chip connectivity, with Ethernet connections to the Host system.

1.2. System architecture

The SpiNNaker architecture is constructed using custom MPSoCs. Each chip (Fig. 2b) contains a bespoke router with full-duplex ports for six external connections and its eighteen ARM9 processor cores. SpiNNaker chips may be interconnected using a 2D triangular mesh wrapped into a torus (Fig. 1). While not biologically realistic, this was chosen as an extensible configuration with a wide choice of node-to-node routing paths, providing high aggregate bisectional bandwidth.

Each ARM core has a connection to the Communications Network-on-Chip (Fig. 2c: Comms NoC), as do the external inter-chip links, with the on- and off-chip networks forming a seamless routed whole. This is facilitated by an *asynchronous* interconnection medium so the *whole* is a GALS (Globally Asynchronous, Locally Synchronous) system [14]. The asynchronous interconnect not only simplifies timing closure, independent of path length, but promises better power economy than a synchronously-clocked alternative with the anticipated network loading patterns.

A second GALS NoC (Fig. 2c: System NoC) gives all cores access to shared peripherals and a separate (in-package) 1 Gb SDRAM (Fig. 2a). This network has quite different requirements: it has to supply sizeable data blocks to the processor cores, usually under DMA control. A shared on-chip 32 KB SRAM – which is used for inter-core message passing communications – is also addressed using this path.

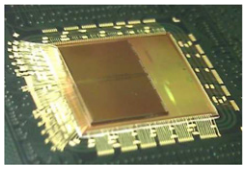
Input/Output is provided primarily by 100 Mb/s Ethernet (Figs. 1 and 2c) provisioned on a subset of nodes. There are also some General Purpose I/O (GPIO) lines which may interface with external devices at a somewhat lower speed. In the future it is further envisaged to use Field-Programmable Gata Arrays (FPGA) chips to interpose non-disruptively into the communication mesh, facilitating high-bandwidth, low-latency sensor/actuator I/O. Further details on the overall SpiNNaker system architecture and its design philosophy to tackle spiking neural computation problems are published elsewhere [19–21,46,45].

1.3. Operation

Neural-network simulation is a highly parallel task; each core is loaded with neural processing software and its neurons' local synaptic weights. After initialisation of all nodes, cores run asynchronously and handle events occurring in real time within the system. Spike packets are routed and replicated in hardware through the machine's communication fabric into multicast trees.

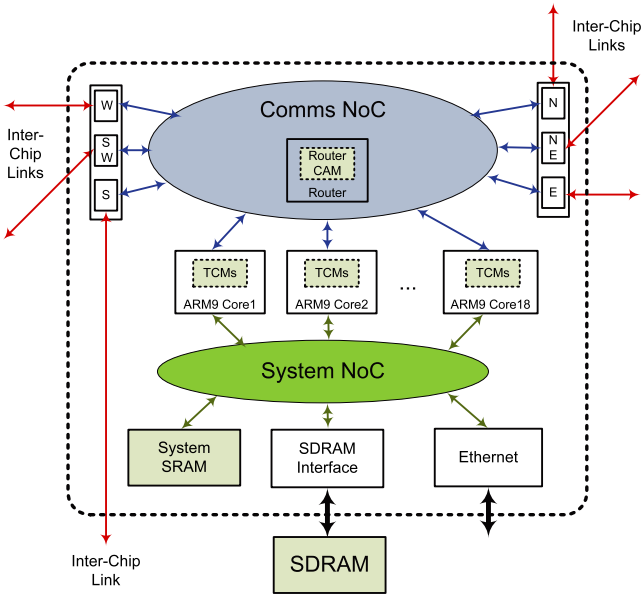
One of the 18 cores on each chip is elected to the rôle of 'Monitor Processor' which coordinates chip-level functions such as non-spike communications, control and management. In order to remain fault-tolerant, this assignment is made dynamically after power-on testing from the set of known-good processors. The remaining cores are assigned as 'Application Processors', and perform neural simulation work. Whilst again this is not a faithful biological replication of the brain, it takes advantage of the electronic medium being used to create the neural network simulations, whilst maintaining the spirit of the *redundancy* and *renewal* inherent to biological systems.

SpiNNaker has been developed primarily as a flexible computing platform to contribute to the scientific Grand Challenge of achieving an understanding of the principles of operation of information processing in the brain. As such it is intended that it be able to support a wide range of different models of biological neural networks, of scales up to a billion spiking neurons, operating in real time at biological firing rates. This target application domain has led to an architecture with unusual properties, most notable the ability to handle very large numbers of very small, independent, multicast packets (up to 10 billion such packets may be sourced and duplicated per second) and this naturally raises the question



(a) A SpiNNaker chip and stacked SDRAM (Photo: Unisem Europe).

(b) Packaged Production SpiNNaker chip.



(c) Simplified schematic of the SpiNNaker chip highlighting its communications architecture.

Fig. 2. The SpiNNaker multi-processor system-on-chip (MPSoC).

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

as to whether there may be applications outside neuroscience that can also benefit from this unusual architectural capability.

It is anticipated that typical customers for SpiNNaker are psychologists, neuroscientists and multi-disciplinary teams using principles from biology to create simulations, as well as others bringing their highly parallelisable tasks to the platform. These users will wish to gain visibility of application software and hardware performance data 'in-flight', thus the communications fabric must also support monitoring and debugging traffic. Rather than provide a separate network for this management information, it is multiplexed onto the same communications fabric as the multicast spike traffic, minimising power requirements and simplifying MPSoC design. The requirements of management services are different from those of spike traffic, but to ensure neural traffic is not disrupted, the alternative packet types retain the same short message length principle. Longer messages and reliable delivery are handled by higher-level software protocols.

The focus of this paper is the layered hardware and software forming the heterogeneous communications networks of SpiNNaker, facilitating its expansion into a very large high-performance computing platform.

2. Similar work

Many different system architectures have been used for neural modelling, particularly High-Performance Computing (HPC) systems and bespoke accelerators. Custom accelerators [26,43], graphic processors [25] and general-purpose accelerators [7] are outside the scope of this paper due to their inability to scale to very large simulations. This section therefore reviews historical and state-of-the-art extensible systems.

In the early nineties a team at U.C. Berkeley worked on the Connectionist Network Supercomputer [5]. This project aimed to build a supercomputer specifically tailored for neural computation. The system was designed to be a 2D mesh, with a target size of 128 nodes (scalable to 512), each incorporating a general-purpose RISC processor plus a vector coprocessor, 16 MB of RAM and a router. This system was very similar to that of SpiNNaker in that its interconnection network was expected to support application, I/O and management traffic. Its architecture included a host machine, directly attached to the mesh interconnect, therefore it did not have the flexibility of being connected remotely. Apart from system scale, an important difference is that it did not support multicast communication; the system was expected to rely entirely on spatial locality to scale to the required performance. A prototype of the node was built under the codename T0, but it is not believed the system operated as a large network. The results of experiments using up to five nodes in a bus configuration were also published [44].

The Blue Brain project [36] and the early stages of SyNAPSE [2] have somewhat similar applications to SpiNNaker. However they do not employ a custom architecture, but use a general-purpose massively parallel system: the IBM BlueGene [23]. This family of supercomputers is composed of tens of thousands of compute nodes, each having several PowerPC processors. In contrast to SpiNNaker, Blue Gene supercomputers have multiple separate, specific purpose networks. The main interconnection network is a 3D torus, which is used only for transmitting application point-to-point traffic. Multicast and broadcast messages are carried through another network and a distinct, Ethernet-based network is used to handle control and I/O traffic. BlueGene/P machines are cited as delivering a peak 379 Mflops/W [39,17] whilst the promised BlueGene/Q in prototype has attained over 2000 Mflops/W [17].

Although there is currently little information publically available about its architecture or operation, Fujitsu's K Computer (heading the November 2011 Top500 list [39] with over half a million cores, and 830 Mflops/W [17]) also appears to follow this philosophy. Preliminary information indicates the existence of a main interconnection network for application traffic [1] and a secondary Infiniband interconnect for I/O and system management. By contrast, the Cray XT family of supercomputers has a similar philosophy to SpiNNaker's of using the main interconnection network for multiplexing application, management and I/O traffic [52,10]. The most power efficient XT5-HE instance achieves around 300 Mflops/W [17].

The ongoing SyNAPSE [51] project is also interesting as it takes a broad multi-pronged approach to neural modelling. Their stated aim to is to 'develop electronic neuromorphic machine technology that scales to biological levels'. This has involved creation of custom hardware [38], using HPC platform modelling techniques (mentioned earlier), and research into exotic areas such as memristor technology.

Finally, the FACETS project [18] is creating a faster-than-real-time custom hardware system for the simulation of large- (but unspecified) sized networks of biologically-inspired neurons. Its distinctive characteristic is that it employs *analogue* circuits to implement neural dynamics. The interconnect however uses digital logic, circuit-switched synchronous communications, for transmitting spikes and supporting system I/O. It is believed that the system management traffic will be served by the same interconnect as for the spike paths—the same philosophy as SpiNNaker.

3. Hardware communication layers

SpiNNaker features interconnection networks at many scales and sizes, from on-chip to inter-chip, to ex-machine I/O connectivity. It is a marriage of various communication protocols: ARM's AMBA [3], asynchronous on-chip 3-of-6 RTZ, chip-to-chip 2-of-7 NRZ and Ethernet. There is a layered approach to connectivity, with standard and bespoke protocols enabling high-performance neural messaging and system management functions to be efficiently supported over all physical transport paths on a large-scale machine.

3.1. Router

The router, embedded within the Comms NoC (Fig. 2c), is a synchronous unit although its clock is independent of others on the chip [53]. It has twenty-four asynchronous full-duplex ports: six for the external connections and eighteen for on-chip processing cores. As the packets are short they are de-serialised and passed through a pipeline (one packet per clock) where they are checked for integrity, processed according to their type and finally passed to an output stage. The last stage attempts to dispatch the packet to its determined destination(s) and will stall if it cannot be output on that cycle. To prevent deadlock, including cases where an output channel may have failed, a programmable time-out applies to this stall so that a packet which gets stuck will eventually be dropped. A dropped packet is retained in a buffer for software examination so it may be reinstated, perhaps with some modification, by the local Monitor Processor. To minimise problems due to serialization delays, there are FIFOs on all outgoing links.

As a further safeguard against both faults and channel congestion on the direct route, the router can attempt 'emergency' routing before dropping a packet. This is a hardware mechanism which sends a packet on a diversion through an adjacent node (Fig. 3). The packet header is modified as it travels so that, if it diverts successfully, it can continue as if nothing has happened when it returns to the original path.

Router inputs need to be sequenced. This is done using an arbitration tree (Fig. 4) which allocates resource on demand, alternating if requests are continuous. The router is capable of

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

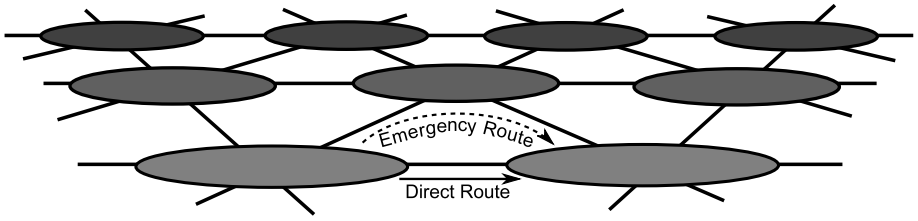


Fig. 3. Emergency routing, an example of the process used when a diversion occurs.

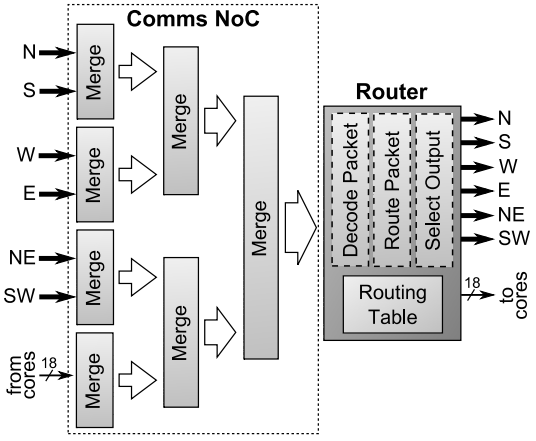


Fig. 4. The asynchronous Comms NoC forms an input arbitration tree for the router.

accepting one packet per clock cycle unless it has backlogged. In normal operation if the router is run at 100 MHz – around half its maximum design speed – a packet will arrive from some source on average every 10 cycles. The asynchronous Comms NoC is efficient during periods where there is no traffic for the router as there is no continuous synchronous clock burning power.

3.2. Packet formats

To cater for the different communication demands, the router supports four types of packets (Fig. 5). Packets are either 5 or 9 bytes in length, comprising an 8-bit header and a 32-bit field typically used for routing, which may be augmented with an optional 32-bit *payload* field. Packet transmission is essentially serial so omitting the payload saves time and increases available packet rate. Example distribution trees can be seen in Fig. 6 for each of the four packet types:

- Multicast (MC): intended for neural spike events. *(1:many)*
- Point-to-Point (P2P): for code distribution and system control. *(1:1)*
- Nearest Neighbour (NN): principally for boot and fault recovery. *(local node:node)*
- Fixed Route (FR): for monitoring and management traffic. *(typically many:1)*.

The control byte (header) for all packet types is similar (Fig. 5). Two bits identify the packet type, one bit indicates the presence of an optional payload and one bit records the entire packet's parity, including any payload. Most packets have a two-bit timestamp which allows routers to drop packets of a certain age, a means to filter 'rogue' traffic caused by faults. The MC and FR packets also have 2 bits of 'emergency routing' information to control routing around a failed or congested link [41] (Fig. 3). The sequence code field for the P2P packets facilitates the structuring of longer messages by higher-layer software protocols.

3.2.1. Multicast packets

Multicast spike packets are distributed to a subset of the neural processors using Address Event Representation (AER) [40,12,11]. For this purpose each node contains an associative routing table consisting of a 1024-entry key, mask and target triplet. Where no match is made from this table the packet 'default routes' to egress opposite its ingress, meaning a table entry is only needed where a packet is steered to destination processing cores, or where it

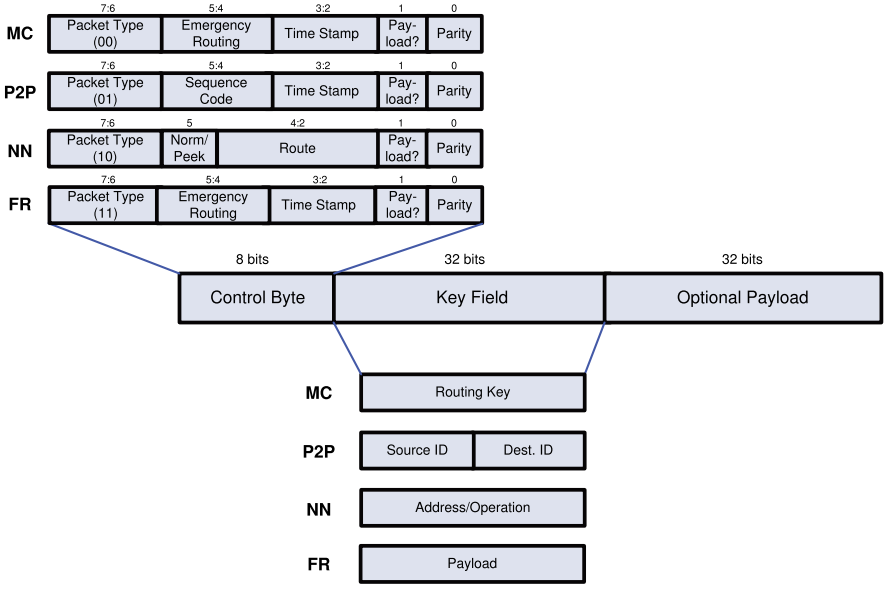
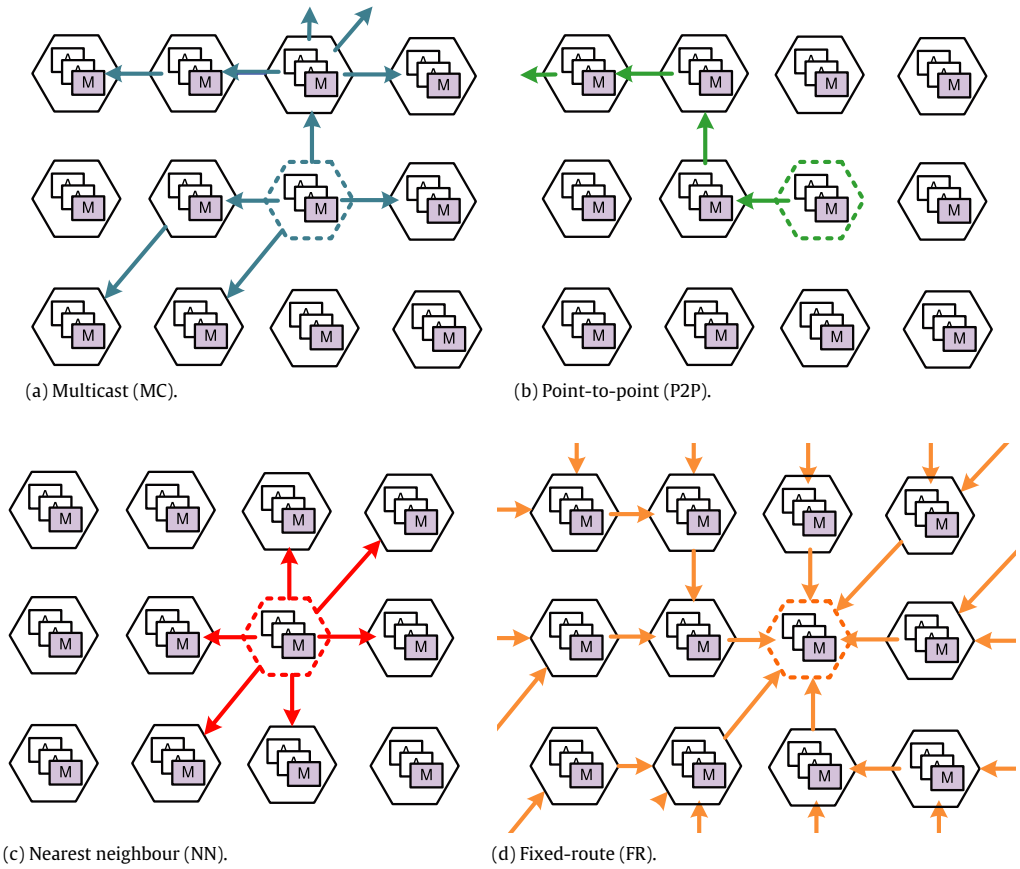


Fig. 5. SpiNNaker packet formats.



(a) Multicast (MC).

(b) Point-to-point (P2P).

(c) Nearest neighbour (NN).

(d) Fixed-route (FR).

Fig. 6. Example distribution trees of each of the four SpiNNaker packet types. Incoming packets are handled by the router, and only interrupt relevant Monitor (*M*) or Application (*A*) Processors as necessary.

needs to turn or bifurcate in transmission (e.g. Fig. 6a). This is a key scalability feature of SpiNNaker communications as the aggregate number of system routing table entries scales linearly with the number of nodes, as does the number of required routing table entries. MC packets may be routed from and to every core in the system. In general they are intended purely as neural spike events and will not carry a payload. They are expected to dominate the network traffic.

3.2.2. Point-to-point packets

P2P packets target a single destination chip, not an individual core, and are delivered to the designated Monitor Processor on the chip. There is a 16-bit destination field which allows system expansion to 2^{16} SpiNNaker chips. At each routing node, the corresponding entry in the 2^{16} P2P entry routing table holds a 3-bit code specifying the direction of the next hop, which is either 'here' or one of the six external links (Fig. 6b). P2P packets are typically used for tasks such as code and data distribution, usually carrying a payload of higher software protocol layers and data.

3.2.3. Nearest neighbour packets

NN packets are used mainly as part of the boot process and for debug access to the neighbouring chips. As the name implies, they are short range, permitting read/write access to a neighbouring chip's shared resources (Fig. 6c). Their distribution tree is addressed by local link ID and not a routing table.

3.2.4. Fixed-route packets

FR packets are similar to MC packets. The difference is that they are routed regardless of source by a single route-word *at each node* so that only fixed, unidirectional merging tree structures can be implemented (Fig. 6d). This packet type typically allows the extraction by Ethernet of monitoring information at low cost both in routing hardware and bandwidth overhead (the 'key' field is available for payload too).

3.3. System NoC

The System NoC provides a path for cores to share chip-level resources (Fig. 2c). It is another GALS NoC but is optimised for different criteria. Its primary goal is conveying large quantities of data amongst its clients, including access to the shared SDRAM, System RAM and peripherals. Typically DMA is used to service high-bandwidth block data transfers, but ad-hoc single-cycle, processor-originated accesses are also permitted—such as for message passing across shared System RAM.

3.4. Ex-system, Ethernet

External I/O connectivity is provided by commodity 100 Mb/s 'Ethernet' links attached to a subset of the SpiNNaker nodes. Chips which detect the presence of a PHY (PHYsical layer transceiver) at power-on enable their Ethernet Controllers, otherwise they remain dormant to save power. The Ethernet is used to transfer spike information in and out of the system, and for system control and management traffic.

4. Software communications layers

The *physical*, system-wide SpiNNaker network is optimised for the expected traffic; primarily it presents a uniform 'flat' source-based routed medium for neural spikes to traverse to their destinations. The secondary load is machine-control traffic, usually from chip-to-chip, handled by the local Monitor Processors.

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

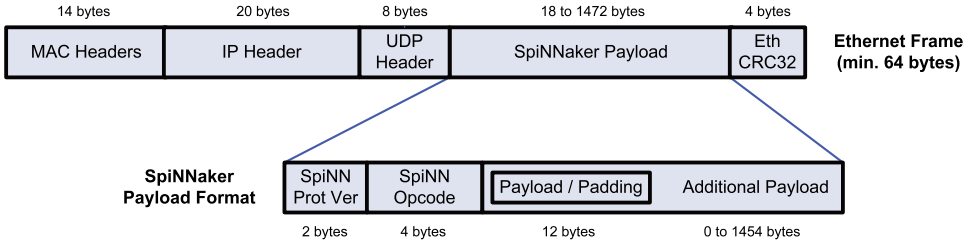


Fig. 7. Ethernet framing format. Data is encapsulated by IP, UDP and SpiNNaker headers.

A need is anticipated for other communication paths, an example of which is providing a path from the host to a core performing neural computations to allow the probing of an individual neuron. This is achieved by overlaying a software protocol on top of the physical hardware and providing a hierarchical communication environment. There are four distinct identified interoperating communication layers:

1. Intra-chip — by the asynchronous Comms NoC and router in hardware. There are four flavours of packets as discussed in Section 3.2. At this layer message passing across the System NoC using the System SRAM for core-to-core non-spike communications is also supported.
2. Inter-chip — the same packet formats, traversing between nodes over the chip-to-chip external asynchronous links (Fig. 1).
3. Ex-system — a connectionless frame format (Fig. 7) provides for connectivity in and out of the machine via the Ethernet connections. In the near future FPGAs are expected to be interposed in the connectivity mesh to be used as an additional interface for the ex-system paths.
4. Internet — beyond the local link, a low-cost connectionless protocol defined in software is used to extend SpiNNaker communications to routed internetworks.

4.1. Ex-system and Internet encapsulation

Internet Protocol (IP) encapsulation of SpiNNaker ex-system data is used to ensure standard networking hardware can handle SpiNNaker-type packets. Using IP facilitates remote access to the machine environment and enables standard 'sockets' programming libraries to be used to interface with the SpiNNaker communications stack.

Transitioning between external and internal machine packets is inefficient as a neural spike datum is small (4 or 8 bytes). To convey this event to a host device it is encapsulated in an Ethernet frame which has a minimum frame size of 64 bytes (Fig. 7). Headers/Trailers for Ethernet framing (18 bytes), Internet Protocol (20 bytes) and User Datagram Protocol (UDP) transport (8 bytes) leave 18 bytes available for the carriage of this payload. Therefore padding must be added, or data aggregated to make full use of resources. UDP was selected as a good match for spikes which are time-sensitive and connectionless, with no facilities for retransmissions. UDP also has a lower overhead cost than TCP, and it requires relatively little implementation effort for minimal functionality. To enable extensibility of the simple SpiNNaker packet format a 2-byte protocol version number and a 4-byte message type/opcode is added (Fig. 7), so that different types of message/instruction can be multiplexed per host/SpiNNaker chip.

4.2. SpiNNaker Datagram Protocol (SDP)

All the layers as described above are *connectionless*, that is they do not store state or make any attempt to detect and retransmit data lost in the transmission process. A protocol called SDP has been created which operates across all of the layers of physical communication, from external host to internal processor (Fig. 8), and provides facilities to transmit datagrams of up to 64 KB in length.

4.2.1. SDP internally within SpiNNaker

SDP allows messages to be sent using sequences of (short) P2P packets inside the SpiNNaker machine (Fig. 9). The traffic flows on the Comms NoC between processors on a chip, and beyond this to processors on another chip via inter-chip links. Each sequence is checksummed and acknowledged, with erroneous and dropped packets identified. This is notified to the application so that it may decide whether a retransmission is to be made. The SDP datagram includes an address and port which can be on *any* core, so SDP can be used to pass messages anywhere; this is achieved by a node's Monitor Processor relaying data via the on-chip shared SRAM across the System NOC (Fig. 8).

4.2.2. SDP outside SpiNNaker

A comparable mechanism is used for external communications where a Monitor Processor on an Ethernet-attached node bridges P2P SDP packets into Ethernet SDP frames (albeit with fewer fragments due to the larger available payload). A transfer from an external host device to an internal processor target is depicted in Fig. 8, where the Ethernet attached SpiNNaker chip acts as a seamless bridge between the internal P2P and the external Ethernet/IP domains for the SDP transport protocol.

5. Power consumption

Power consumption is clearly an issue in any high-performance computing system and is receiving significant attention in the HPC community. SpiNNaker, a specialised HPC architecture, is designed to minimise power consumption while providing considerable processing capability. This was a significant consideration in the selection of processors: ARMs are a typical choice for embedded equipment where low power is desirable, the price typically being individual poorer single-thread performance. In a massively-parallel system the power/performance balance is currently tilted in favour of using many, simple processors [21], especially as those processors may be in a very low-power 'sleep' state whilst inactive.

Power consumption may also be reduced by high integration levels, thus each node in a SpiNNaker machine is based on an MPSoC with few off-chip connections. Around half the off-chip signals are to the SDRAM which shares the same package to reduce capacitance (Fig. 2a). The other off-chip signals are largely the inter-chip communication links and, as the majority of the power dissipation in a CMOS circuit is dynamic, these use asynchronous Non-Return-to-Zero (NRZ) signalling to minimise the number of voltage *transitions*.

Each unidirectional link comprises eight wires: seven carry a data symbol and the eighth is a handshake return. To send a symbol

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

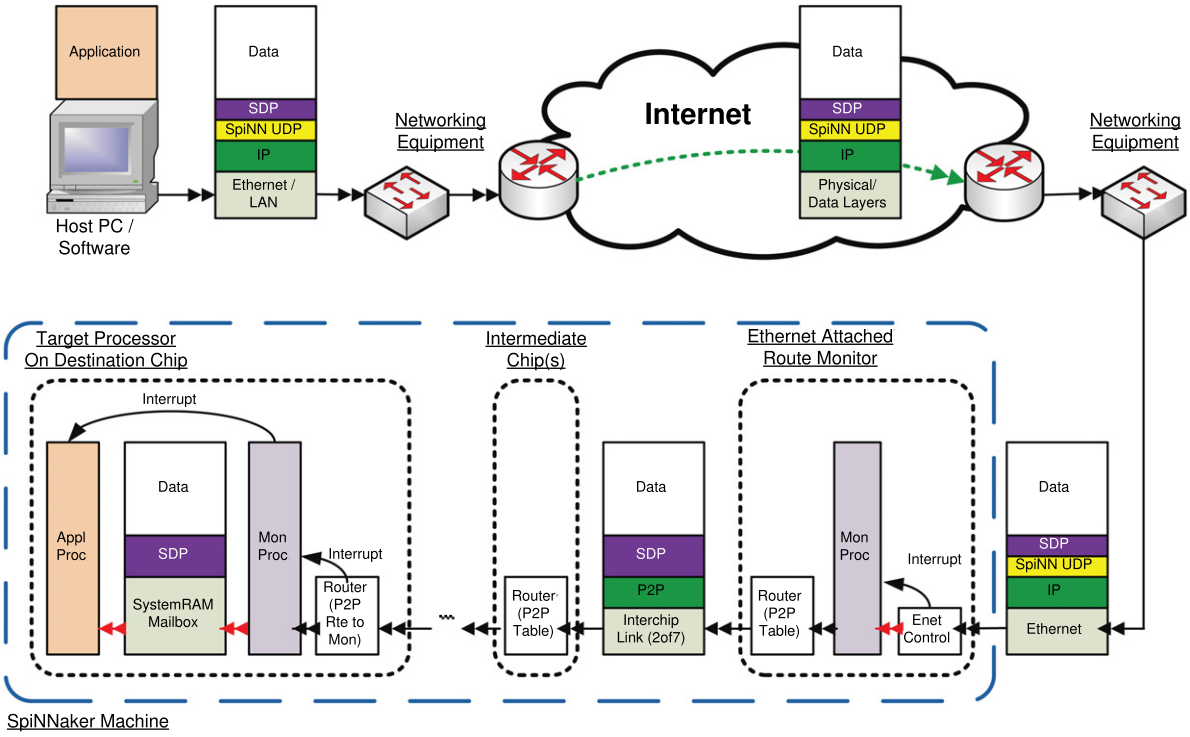


Fig. 8. Communication layering diagram. Data flows from the Host Application (top left) across all 4 distinct communication layers to the destination SpiNNaker Application Processor (bottom left).

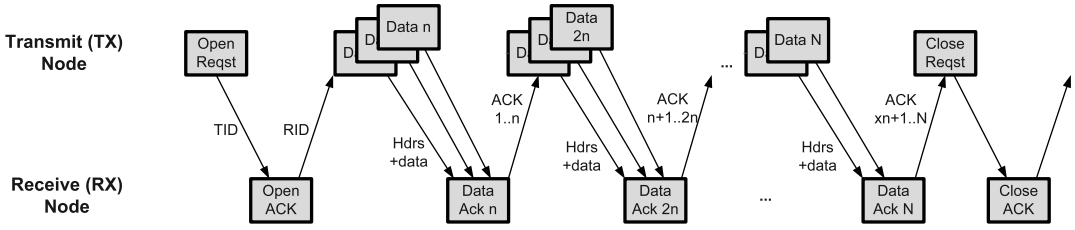


Fig. 9. An SDP data flow over P2P from transmitting chip to receiving chip.

the transmitter makes transitions on two of its seven outputs then waits for a single handshake return. There is no return-to-zero phase; the next symbol can be sent immediately. There are 21 possible 2-of-7 codes of which 17 are valid here: thus four bits of data or a packet end marker are sent in one symbol [6].

The cost of sending a bit between chips depends on whether or not a payload is appended, but is around 0.8 transitions. This is similar to an 'asynchronous' serial line (average 0.75 transitions/bit), another technique used for short message transmissions. Transition cost/bit can be lower in bulk (e.g. Non-Return-to-Zero Inverted (NRZI) gives 0.5 + transitions/bit) but carries a synchronization overhead which makes it expensive for short packets.

Keeping power consumption low eases the deployment of a large system, simplifying supply and cooling requirements. It also reduces running costs which, for a high-performance computer, form a significant proportion of the total cost of ownership.

6. Software and data distribution

Software to cover the boot sequence from power-on through to application execution is already in service. At present valuable experience is already accruing with practical (hardware) experience of small (~100 core) networks. It has been noted some of the existing *neural network to SpiNNaker* mapping software will not scale to the largest systems, and will require development as the hardware implementations gain in scale.

Significant challenges are associated with the time taken to distribute software – and, particularly, configuration data – throughout the network prior to application execution. To keep the load-time start latency within bounds, both commonality and parallelism will need to be fully exploited.

A further issue for the mapping software is that it must be capable of dealing with various component failures as these are highly probable in large systems. Indeed, for economic purposes, it is intended that in the SpiNNaker architecture most faulty chips will be pressed into service as the majority of faults are likely to affect only a single core. The faulty core is isolated during power-on self-tests, leaving the remaining 17 processors in operation for Monitor and Application rôles.

6.1. System boot

Following a power-on reset, all processors independently initialise and test their hardware using an on-board ROM image. They then enter a quiescent state, except for the Monitor Processor which listens on the inter-chip links and the Ethernet interface if it is enabled. The host system then injects a software image via Ethernet which is assembled and executed on one or more connected nodes.

This software image then propagates itself through the network as a flood-fill (Fig. 10) using Nearest Neighbour communication—the only available mechanism until nodes are numbered and

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

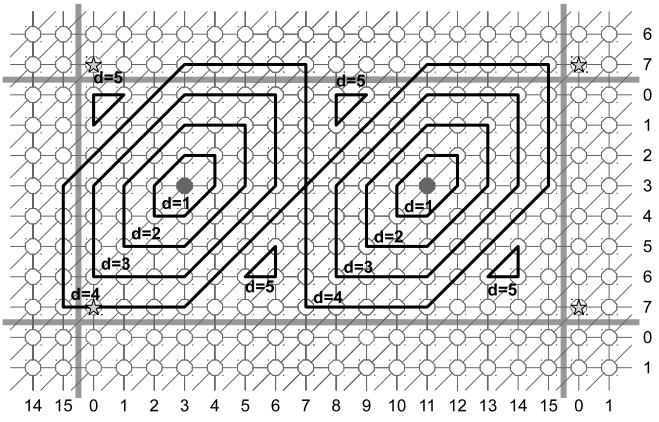


Fig. 10. 'Waves' of system boot flood-filling a tessellated SpiNNaker torus. The two dark nodes represent the two Ethernet attached chips seeded by the host, the star represents the origin of coordinates node.

routing tables populated. The use of self-propagating flood-fill greatly reduces the time taken for system boot as this is performed in parallel rather than sequentially. In a rectangular $M \times N$ network (M being the *smaller* dimension) the diameter (D) or hop-count from any one node to its most distant is:

$$D = \left\lfloor \frac{N}{2} + \max\left(0, \frac{2M - N}{6}\right) \right\rfloor. \quad (1)$$

The use of well-placed multiple software seeding points reduces this distance. In the example of Fig. 10, the one seed diameter is 8 hops, but using two seed nodes reduces the distance to 5 flood-fill hops.

Experiments on SpiNNaker hardware using this self-propagating technique yielded results shown in Fig. 11a, further detailed in [50]. The times, as expected, proved linear to the number of hops, and the duration of the system boot phase for any regular topology can be anticipated (Fig. 11b).

The topology of SpiNNaker machines is expected typically to be 'square'. A maximal 2^{16} node system therefore (using Eq. (1)) has a distance of: $D = 170$ hops from a single code injection point which suggests a maximum (32 KB image) system boot time of 5.3 s (from Fig. 11b). Usually the time taken will be smaller, e.g. around 4 s for a 24 KB system boot image.

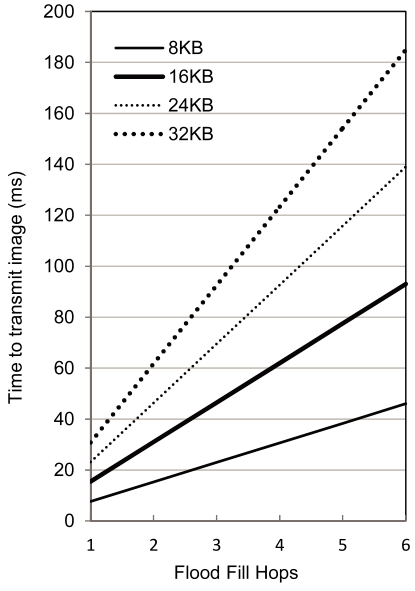
If the topology and relative positions of the various injection points are known, this phase can also number the nodes. Once the nodes are numbered, point-to-point routing is possible and the host can communicate with any node directly. At this stage the host will additionally collate any fault-reports from nodes to ascertain the health of the system for use in the mapping stage before distribution of code and data.

6.2. Application loading

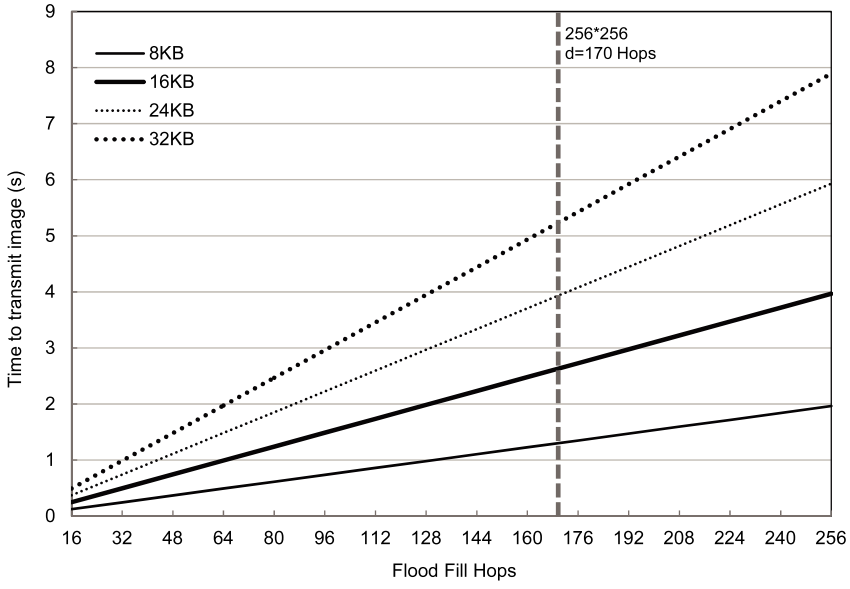
Following system boot, an application can be loaded serially onto each core using SDP. This method, as used by the current small systems, lacks parallelism and the time taken scales linearly with the number of nodes. Thus a flood-fill mechanism will also be used for application distribution, exploiting the fact that most cores will be running identical software. Monitor Processors will have a different image from Application Processors, but that too can be replicated across the machine using flood-fill.

With a booted and mapped system the neural application can be 'place-and-routed'. This involves placing *populations*, groups or individual neurons on specific chips and cores for simulation. At the small scale this task is performed on the host. In the longer term this will become infeasible because, although code may be identical, the synaptic data-set for each node is unique and the data is *much* larger than the code. Serial loading of ~8 MB to 10^6 application cores is clearly not practical: with a single 100 Mb/s Ethernet link the load time would be measurable in *days*. More links can be added to reduce the transmission time, but a larger problem is faced by the host, as the neuron place-and-route of a very large neural network is a compute-intensive NP-complete problem [13,24].

However, users do not specify each neuron and connections individually: these are generated statistically from the user's rules. It is feasible that these aggregated smaller data sets can be initially distributed coarsely, with SpiNNaker itself performing the detailed place-and-route. There are still many unsolved problems in this space, including derivation of the neural MC routing tables based on the desire for tightly connected neurons to be located proximately. Work on these challenges is ongoing.



(a) Small 6-hop test system.



(b) Extrapolated large system.

Fig. 11. Flood-filling the system boot image. In a 256 × 256 system, the maximum distance is 170 hops.

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

6.3. Run time

When an application is running the network activity can be divided roughly into two. It is expected that application traffic will dominate and the majority will be neuron spike events. It is likely that, in the longer term, some spikes will be routed directly to and from I/O devices which will chiefly traverse the Ethernet, and FPGA interfaces in the mesh. Current (small system) I/O is Ethernet SDP traffic to and from the host.

The other class of run-time traffic is for machine monitoring and control. Statistics such as core and network utilisation, packet error count and chip temperature may be requested from each node and made available to the host. These requests are optional, and typically would be used more comprehensively during researching new network configurations. Primarily these 'management' requests will be handled by Monitor Processors although SDP allows the probing of every core directly.

Other control traffic will occur without user intervention using Nearest Neighbour communications. This permits system status – such as fault reports – to disseminate through the machine. Eventually it is planned to attempt some self-repair by altering routing around faulty devices dynamically. With a million processors available, the policy will be to leave some spare cores available to cover for those with faults.

7. Results

Full-scale SpiNNaker silicon has been available since May 2011. Naturally, initial hardware tests concentrated on proving its functionality but some measurements to quantify network bandwidth, latency and power consumption have also been made.

7.1. Communications bandwidth

Without a large network the router must be stress-tested by generating packets from local cores. Here it is possible to achieve 5.3 Gb/s at entry to the router, shared amongst the cores by the input processor merge tree (Fig. 4). In operation the *local* demand will be much less than this. A single (overclocked) core in a tight loop can individually achieve 5.0 Gb/s.

Each off-chip link is capable of sustaining around 250 Mb/s (1.5 Gb/s in aggregate), a figure limited by the on- and off-chip delays. This is disappointing as it represents only ∼6 million spikes per second per link. This would be too slow to support 10^9 uniformly distributed neurons firing at 10 Hz across a section of the system but, as in biology, spatial locality will limit transit traffic drastically. (Local neurons connect with high density to neighbours, and much more sparsely afar) [42]. In practice the available bandwidth is still several times the expected system requirement; now closer to 40% than the originally anticipated 10% [41].

Communications bandwidth is more important to the on-chip System NoC for fetching (synaptic weight) data from the SDRAM. This achieves 5.6 Gb/s when employing burst traffic via DMA—the expected method of access. It is anticipated that this will meet the requirements of the neural applications (16 M synapses × 10 Hz × 4 bytes). The on-chip SRAM has an independent narrower connection and achieves 3.2 Gb/s in bursts, or around 200 Mb/s one-word-at-a-time. Both methods may be used at run time as appropriate (e.g. message passing, block transfers), but the demands will not be high.

7.2. SDP

In the current small machines, SDP is used to load application data and executable code. A measured test, transmitting data from a host machine to an Ethernet attached SpiNNaker chip and then to an Application Processor on that node via shared memory, achieved speeds in excess of 22 Mb/s. With the target Application Processor on a different chip the scenario now appears as per Fig. 8. Here the payload transmission speed is 5.3 Mb/s (around 4.5 μs per packet), due to the fragmentation, bridging and acknowledgements of the internal SDP using P2P packets (Fig. 9).

The SDP datagram sizes used in the tests were 256 bytes. The total Ethernet frame length of an encapsulated 256 byte SDP packet is 328 bytes: 18 bytes of Ethernet Headers/Trailers, 20 of IP, 8 of UDP and 26 of SDP giving an overhead of 72 bytes. This is an efficiency of 78%. For 100 Mb/s Ethernet therefore, the maximum potential data rate using SDP is 78 Mb/s.

SDP is experimental and yet to be optimised; for example the host will not send a new Ethernet frame until the previous SDP packet has been acknowledged. This current limitation means the round trip delay time of perhaps 0.1 ms needs to be taken into account. As there are ∼10, 000 frames per second where the rate currently peaks at 22 Mb/s, this could go some way to explaining the current sub-optimal performance.

Internally to SpiNNaker, P2P packets are used to convey SDP data. Each 72-bit SDP P2P packet carries a 24-bit payload. After including the headers this gives an efficiency of 31%. Given the 250 Mb/s chip–chip link rate this suggests a peak data rate ∼80 Mb/s is possible, matching that of the Ethernet.

A latency estimate for the internal links is ∼400 ns per hop, comprising around 300 ns for serial transmission plus 100 ns from the router pipeline, assuming no congestion. A similar latency applies to the datagram (not *packet*) acknowledgement. This represents ∼1 μs per hop for each 256-byte datagram on top of the ∼48 μs transmission time; small for a few hops but becoming significant over long distances.

7.3. Assessment of network latency constraints

Maintaining network latency below the resolution of the real-time biological application (typically 1 ms) is an important aspect of SpiNNaker. If it is larger than this it will be added to the application-imposed latency, adversely affecting application semantics.

The simulation framework described by Navaridas et al. [41] has shown how latency scales with system size by simulating a range of system configurations from 16 × 16 (256 nodes) to 256 × 256 (65,536 nodes). In all cases the systems are dealing with the expected maximum load generated by the biological application [41]. Two spatial traffic patterns are used. A *local* distribution emulates the way neurons communicate: most traffic destinations are in close proximity. Traffic following a *uniform* distribution represents a very pessimistic scenario in which locality is not exploited.

The average and maximum latencies obtained after a simulation of 100 kcycles are plotted in Fig. 12. The experimentally measured silicon link speeds and latencies have been used as parameters in this simulation. End-to-end latency scales linearly with the number of nodes per dimension, i.e. $O(\sqrt{N})$ with the number of nodes. The maximum latencies for the two traffic patterns are very similar because it is the latency needed to reach the most distant nodes, the differences being attributable to greater network contention. However, when traffic is largely local the average delay is barely affected by network size whereas with uniform traffic, latency is noticeably higher. Network latency is around an order of magnitude smaller than application resolution and therefore should not interfere with application semantics. Network latency, therefore, is not expected to be a limiting factor of SpiNNaker's real-time scalability.

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

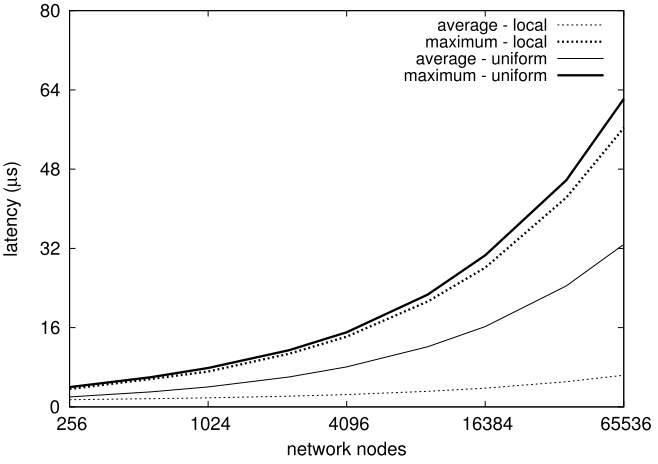


Fig. 12. Latency scalability under uniform and local traffic distributions.

7.4. Power

Each SpiNNaker chip, while Application Processors are quiescent, draws 360 mW. Each processor running at 150 MHz adds 18 mW, therefore with all 18 processors running a SpiNNaker chip consumes 680 mW, or close to 900 mW at 200 MHz. In a 2^{16} node system this is ∼60 kW of *core* power. This is a *peak* figure; in neural applications it is anticipated that processors will be 'asleep' for a significant proportion of the time.

- ARM9 delivers 1.1 Dhrystone MIPS/MHz [4] thus achieving $\frac{18 \times 200 \times 1.1}{0.9} = 4400$ MIPS/W.
- The theoretical peak processing performance of a SpiNNaker is: $2^{16} \times 18 \times 200 \times 1.1 = \sim 250$ TIPS.

At full speed the cost of running an *inter-chip link* is 6.3 mW. Therefore the consumption per bit, is 2.5×10^{-11} J, or possibly more clearly: ∼40 Gb/J. With six outputs per chip and an assumed 40% utilization this adds 15 mW to each chip's consumption, or around 1 kW to the total for a 2^{16} node system. When the SDRAM is continuously exercised (at 130 MHz clock) the node power requirement increases by 170 mW representing a further 11 kW in a fully expanded system, proportionally less if the SDRAM duty cycle is <100%. The only other components of the system are a few active Ethernet ports (∼40 mW each), LEDs etc, which consume a low amount of power relative to the rest of the system.

The overall peak system power should be in the range 30–75 kW, depending on activity for a maximally configured 2^{16} node SpiNNaker system, with a system efficiency bettering 3500 MIPS/W.

8. Conclusions and further work

SpiNNaker is an unconventional, massively-parallel, high-performance architecture. It is biologically inspired and exploits the properties of electronic computers to build a simulator for biological networks.

The current state is that small-scale demonstrators now exist and both the hardware and software for larger systems is under construction; large system models for traffic estimation and analysis have existed for some time.

The chip appears fully functional. There are some minor disappointments: primarily the chip-to-chip bandwidth is less than intended due to large off-chip delays. This pad delay was apparent in simulation and the logic was optimised accordingly, but this was the best the process could achieve. The delays are compounded by the asynchronous nature of the inter-chip links, where 2-way handshaking is required for each symbol transmission. However, the current link speed is still expected to be adequate at all scales as a considerable margin was built in. The 'quiescent' chip power, although acceptable, is also slightly higher than was expected. It is proposed to manage the situation by employing lower clock rates for idle chips/cores, proportionately reducing their dissipation.

An active chip dissipates ∼1 W — less if, as expected, the processors can 'sleep' for significant periods. This suggests that a large-scale system is feasible without resorting to 'exotic' cooling. The communications traffic at its peak will consume on average less than 3% of the peak power per chip. The indications also show that the communications architecture will scale to its nominal limit of a million-plus processors — although there is significant scope for other limitations to become apparent as the research into real-time massive neural network simulation progresses.

Employing GALS interconnectivity has made interconnection straightforward, both on- and inter-chip. Test PCBs have been interconnected by cables to build systems of 16 and more chips without problem. The 2-of-7 coding is efficient with power and has reasonably good bandwidth, although one disadvantage is the need for high pin-count connectors. There is a case for faster, more conventional serial connections although the technology for this on the MPSoC was not readily available.

The SDP layer which was developed to overlay the heterogeneous networks in the machine is successfully in use to distribute code and data in small SpiNNaker systems. Now that this functionality has been proven, it requires optimisation towards the physical limits of the SpiNNaker communications fabric.

It has been demonstrated that it is possible to simulate a thousand neurons per core [32], and create a 4-chip, 64-core (16 Application Processors per node) real-time neural network simulations on the new silicon.

In comparison with rival systems, SpiNNaker lies between the generalised, supercomputer solutions and hardware solutions such as FACETS. It provides more flexibility than FACETS in that the neural models are software programmable but, for the same reason, it cannot approach the same simulation speed. Its capital cost should be less than machines such as BlueGene, and while it does not provide facilities such as floating-point hardware, it is believed that fixed-point calculations are adequate for neural simulation, a task at which ARM processors excel with high energy efficiency. Finally it is a way to connect a massively-scalable system from 18 to a million processors which can easily be expanded as required by the user.

8.1. Further work

It is planned to scale SpiNNaker systems to 10^3 processors by early 2012, 10^4 processors by mid-2012, and to the million-plus processor machine by the end of 2012. Design and fabrication work has begun on hardware to support large numbers of SpiNNaker chips and create the desired topologies. For the interconnections this includes concentrating a number of GALS links for inter-PCB fast serial connection using FPGAs, which also permits low-latency I/O access directly into the mesh.

The biggest current obstacle is the distribution of neural data in large networks. Placing-and-routing on a host and subsequently downloading the data is adequate for small networks, but as the network grows this leads to unacceptable host compute and data transmission times. It is therefore essential to distribute this load into the SpiNNaker machine itself.

The purpose of the simulator is not just to 'build a brain' but also to monitor and understand how it works. The network has considerable (optional) provision for probing operation both in real time and post-mortem, and this will be explored in future research.

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

The scalable SpiNNaker architecture is also garnering interest for use in applications beyond the purely spiking neural space. In addition to Izhikevich [32] and LIF [48] spiking models, Multi-Layer Perceptron (MLP) networks [33] have been built. One institution is already taking advantage of SpiNNaker machines to run distributed ray-tracing applications, and others are actively collaborating in applications where a massively-parallel system would be beneficial.

Finally, although the hardware specification for this generation of silicon is fixed, there is work required on the software communications layer, in order to take full advantage of the hardware's native speed for data and code distribution. The number of Ethernet connections to be deployed to support the real-time I/O and monitoring requirements of a system also needs to be determined.

SpiNNaker is a scalable massively-parallel architecture with a vast, many-layered, heterogeneous communications architecture whose performance is fundamental to the operation of the system. The results from the initial silicon presented in this paper give us considerable optimism that the system will scale to its million-plus processor target.

Acknowledgments

The SpiNNaker project is supported by the Engineering and Physical Sciences Research Council of the UK, through Grants EP/D07908X/1 and EP/G015740/1, and also by ARM and Silistix. Cameron Patterson is supported by a Doctoral Training Award funded by the UK Medical Research Council. Dr. Javier Navaridas is a Royal Society Newton International Fellow. The authors appreciate the support of these sponsors and industrial partners.

Appendix. Neural computation

The biological brain is a massively-parallel computational organ. A brain out-performs any electronic computational platform by several orders of magnitude in energy efficiency, and has the remarkable ability to continue to operate effectively even whilst impaired.

However, neuroscience still has a key obstacle in understanding the brain—its work is typically invasive and destructive. Research into simulating large, biologically-plausible neural networks (brain-like systems) has remained a highly active area over many decades. Goals of this work include creation of realistic simulation platforms to use for experimentation, and to unlock the intrinsic parallelism and efficiency of the brain in order to exploit nature's design in the creation of faster and more efficient computing platforms.

A.1. Neural network modelling

The field of Artificial Neural Networks (ANNs) spawned from the 1943 work of McCulloch and Pitts [37] in proposing the Threshold Logic Unit (TLU) to model artificial neurons. In a TLU neuron inputs are weighted and summed then compared to a simple threshold value to determine whether their output is a binary 0 or 1. However this approach turns out to be limiting, and not particularly biologically faithful. The problems with this model led in part to the creation of a more flexible second generation of ANNs [9], which are also capable of 'learning'. The output from a second generation artificial neuron is no longer binary, but a position on a continuous (typically sigmoid) activation function. Here, the output position represents the frequency of output spikes emitted by the neuron. These *rate-coded* ANNs are today commonplace, but there are classes of problem that prove intractable using this technique [34]. This, in turn, has led to the development of a third generation of ANNs—Spiking Neural Networks (SNNs) [34]. SNNs are not based on the rate of a spike train, but on the *temporal* information encoded by spike arrival times, and thus are more biologically accurate. Crucially, Maass [35] has proven that for sigmoidal activation functions, SNN networks can act equivalently to rate-coded ANNs, which can also model TLUs.

Similarly to neurons, there are multiple synaptic models available for use in simulation [16]. Synapse models are an important part of neural network modelling, learning and plasticity, transforming spikes from afferent neurons into electrical stimuli delivered to the input dendrites of downstream neurons.

A.1.1. Model fideblity

In some neuron models, such as those developed by Izhikevich [30], there is almost complete abstraction from the biology to the computational model, covering only the most fundamental neural dynamics. Conversely, other models types such as Hodgins & Huxley [28] (used in Blue Brain [36]), are computationally complex, but are much more biologically accurate. Adding biological fidelity typically involves reducing the size of network that may be created, or the speed at which it may be simulated on a given platform. Different projects and protagonists have differing priorities and approaches to this problem [31], trading off different characteristics as required.

A.1.2. Neural network simulation on SpiNNaker

SpiNNaker is a general-purpose programmable architecture targeting spiking neural network modelling. This architecture provides users flexibility to select their neuron and synaptic models, and network interconnection strategy. SpiNNaker's typical approach is to deliver SNNs of a real-time flavour, and its interconnection networks are sized accordingly [42]. If the models used are more complex, there is opportunity to reduce the number of neurons per processor. If greater timing resolution is required then this may be configured, up to and including the flexibility to drop out of real-time mode. Further details on neural models and principles are beyond the scope of this paper and readers are directed to the included references, and to review papers from authors such as Bishop [9].

A.2. Results from a SpiNNaker neural simulation

Test boards with 4 SpiNNaker chips (72 cores) have been available since May 2011 and continue to undergo hardware and software testing. The example presented here is of a set of 16 self-connected populations, each producing a self-perpetuating cascade of 'looped' spike trains, the spike rate being a function of the applied bias current. PyNN [15] was used as the high-level specification language for the network populations and connection strategy, which is mapped and optimised by the SpiNNaker compilation tool-chain [22] into binaries for core and chip-level memories. There are 16 populations of 256 Leaky Integrate and Fire (LIF) [48] neurons, each assigned to a separate application core. The output is plotted in real-time to a user's screen for monitoring (Fig. A.13).

In the left panel of Fig. A.13, differing colours/tones represent the average *firing* rates of the neurons in each population. It is possible to interactively influence the population spiking rates in real-time from outside the network by increasing or decreasing the bias current applied to all neurons within that population (in the example core/population (3, 1) is selected).

In the right panel of Fig. A.13 the user has initiated another real-time plot window from the GUI where they have 'zoomed' into an individual population and can examine the firing pattern of all its constituent neurons. This plot scrolls in real-time, and as changes are made over the 10 s period, the user sees the results of a reduction in the bias current (7m8 → 7m10), followed by a gradual increase (from 7m10 onwards).

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

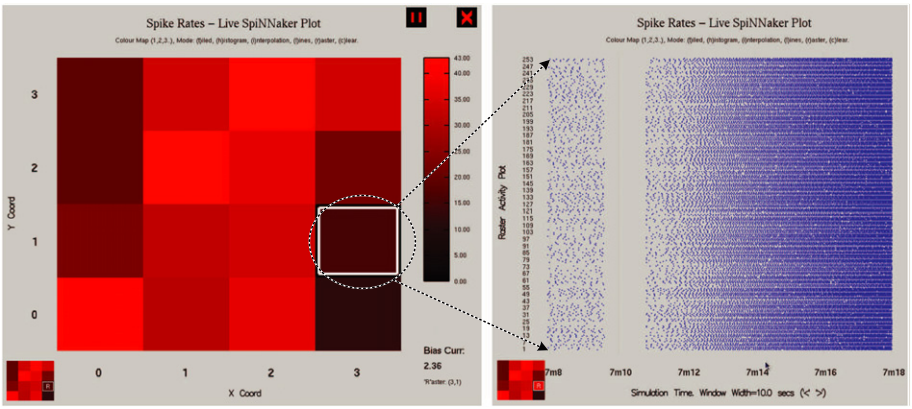


Fig. A.13. Interactive real-time neural network plots on SpiNNaker. Left: population rates (average spikes/neuron/second), users can adjust bias currents for their chosen population e.g. (3, 1), and may open a second raster plot window (right) to view its individual neuron firing events over time.

Table A.1
Measured spike and packet rates across the SpiNNaker communications network for the experiment in Fig. A.13. Transmit (TX) spikes are measured by the originating population core, and receive (RX) rates by the Ethernet attached host system running the real-time visualisation software.

Simulation time	7m8	7m10	7m12	7m14	7m16	7m18	*Sim Max*
Bias applied (mA)	0.75	≤ 0.60	1.01	3.12	4.41	5.26	≥ 28.26
TX spikes/population/s	1602	0	3057	11,575	15,973	18,727	64,000
RX SDP Ethernet frames/s	843	0	999	978	1035	1060	1775
RX av. spikes/neuron/s	6.3	0.0	11.9	44.9	62.2	72.6	250.0

The communications path used to provide real-time visibility of the spikes begins with SDP packets traversing from the (3, 1) application core to the Monitor Processor. The data is then bridged towards the host visualiser via the Ethernet attached chip. This path is similar to Fig. 8, without the intermediate nodes. The rates of real-time spike data are seen in Table A.1, at various stages of the transmission process.

The host receives up to the maximum 64 K spikes per second corresponding to all 256 neurons of the population spiking at their maximum rate of 250 Hz (there is a 4 ms refractory period). Each Ethernet SDP packet may contain up to 64 spikes/256 bytes of data, therefore the frame count only begins to rise significantly once they fill and more are required. In this simple neural connectivity experiment, it is verified that it is possible to transmit the necessary 64 K spikes each second in real-time from a SpiNNaker core to an external host device across the layered heterogeneous communications path.

A.2.1. Other SpiNNaker simulations

Several software models are actively being worked on, including various SNN models (and also perceptron and non-neural applications). Earlier SpiNNaker experiments operated with 2-core test silicon, whose results remain valid for the full 18-core chip, and can be scaled onto the new larger systems [32,48,49,47]. A modular 48-chip board is scheduled for introduction in early 2012, together with correspondingly larger simulations.

References

[1] Y. Ajima, S. Sumimoto, T. Shimizu, Tofu: a 6d mesh/torus interconnect for exascale computers, Computer 42 (2009) 36–40.
[2] R. Ananthanarayanan, S.K. Esser, H.D. Simon, D.S. Modha, The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC'09, ACM, New York, NY, USA, 2009, pp. 63:1–63:12.
[3] ARM, AMBA open specifications, http://www.arm.com/products/system-ip/amba/amba-open-specifications.php, 2011, [Online; accessed 12-Dec-2011].
[4] ARM, Dhrystone and MIPs performance of ARM processors, http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka3885.html, 2011, [Online; accessed 12-Dec-2011].
[5] K. Asanovic, J. Beck, J. Feldman, N. Morgan, J. Wawrzynek, A supercomputer for neural computation, Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN), vol. 1, pp. 5–9.
[6] W. Bainbridge, W. Toms, D. Edwards, S. Furber, Delay-insensitive, point-to-point interconnect using *m*-of-*n* codes, in: Ninth International Symposium on Asynchronous Circuits and Systems, 2003, Proceedings, pp. 132–140.
[7] M. Bhuiyan, V. Pallipuram, M. Smith, Acceleration of spiking neural networks in emerging multi-core and GPU architectures, in: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW), pp. 1–8.
[8] T. Binzegger, R.J. Douglas, K.A. Martin, Stereotypical bouton clustering of individual neurons in cat primary visual cortex, Journal of Neuroscience 27 (2007) 12242–12254.
[9] C.M. Bishop, Neural Networks for Pattern Recognition, first ed., Oxford University Press, USA, 1996.
[10] A. Bland, J. Rogers, R. Kendall, D. Kothe, G. Shipman, Jaguar: the world's most powerful computer, in: Cray user group 2009, Cray Inc.
[11] K. Boahen, Communicating neuronal ensembles between neuromorphic chips, in: T.S. Lande (Ed.), Neuromorphic Systems Engineering, in: The Kluwer International Series in Engineering and Computer Science, vol. 447, Springer, US, 1998, pp. 229–259.
[12] K. Boahen, Point-to-point connectivity between neuromorphic chips using address events, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 47 (2000) 416–434.
[13] S.H. Bokhari, On the mapping problem, IEEE Transactions on Computers 30 (1981) 207–214.
[14] D.M. Chapiro, Globally-asynchronous locally-synchronous systems, Stanford University, CA, Ph.D. Thesis, 1984.
[15] A.P. Davidson, D. Brüderle, J.M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, P. Yger, PyNN: a common interface for neuronal network simulators, Frontiers in Neuroinformatics 2 (2009) 1–10.
[16] P. Dayan, L.F. Abbott, Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems, first ed., The MIT Press, 2001.
[17] W. Feng, K. Cameron, The green 500 list: environmentally responsible supercomputing, http://www.green500.org/, 2011, [Online; accessed 12-Dec-2011].
[18] J. Fieres, J. Schemmel, K. Meier, Realizing biological spiking network models in a configurable wafer-scale hardware system, in: IEEE International Joint Conference on Neural Networks, 2008, IJCNN 2008, (IEEE World Congress on Computational Intelligence), pp. 969–976.
[19] S. Furber, A. Brown, Biologically-inspired massively-parallel architectures—computing beyond a million processors, in: International Conference on Application of Concurrency to System Design, pp. 3–12.
[20] S. Furber, S. Temple, Neural systems engineering, Journal of the Royal Society Interface 4 (2006) 193–206.
[21] S. Furber, S. Temple, A. Brown, High-performance computing for systems of spiking neurons, in: AISB Workshop on Grand Challenge 5: Architecture of Brain and Mind, pp. 29–36.

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

[22] F. Galluppi, A. Rast, S. Davies, S. Furber, A general-purpose model translation system for a universal neural chip, in: K. Wong, B. Mendis, A. Bouzerdoum (Eds.), Neural Information Processing. Theory and Algorithms, in: Lecture Notes in Computer Science, vol. 6443, Springer, Berlin/Heidelberg, 2010, pp. 58–65.
[23] A. Gara, J. Moreira, IBM Blue Gene supercomputer, IBM Research Report, 2011.
[24] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979.
[25] B. Han, T. Taha, Neuromorphic models on a GPGPU cluster, in: The 2010 International Joint Conference on Neural Networks (IJCNN).
[26] H. Hellmich, M. Geike, P. Griep, P. Mahr, M. Rafanelli, H. Klar, Emulation engine for spiking neurons and adaptive synaptic weights, in: 2005 IEEE International Joint Conference on Neural Networks, 2005, IJCNN'05, Proceedings. vol. 5, pp. 3261–3266.
[27] S. Herculano-Houzel, The human brain in numbers: a linearly scaled-up primate brain, Frontiers in Human Neuroscience 3 (2009).
[28] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, The Journal of Physiology 117 (1952) 500–544.
[29] E.M. Izhikevich, Polychronization: computation with spikes, Neural Computation 18 (2006) 245–282.
[30] E.M. Izhikevich, Simple model of spiking neurons, IEEE Transactions on Neural Networks 14 (2003) 1569–1572.
[31] E. Izhikevich, Which model to use for cortical spiking neurons?, IEEE Transactions on Neural Networks 15 (2004) 1063–1070.
[32] X. Jin, F. Galluppi, C. Patterson, A. Rast, S. Davies, S. Temple, S. Furber, Algorithm and software for simulation of spiking neural networks on the multi-chip SpiNNaker system, in: The 2010 International Joint Conference on Neural Networks (IJCNN).
[33] X. Jin, M. Luján, L.A. Plana, A.D. Rast, S.R. Welbourne, S.B. Furber, Efficient parallel implementation of multilayer backpropagation networks on SpiNNaker, in: Proceedings of the 7th ACM International Conference on Computing Frontiers, CF'10, ACM, New York, NY, USA, 2010, pp. 89–90.
[34] W. Maass, Networks of spiking neurons: the third generation of neural network models, Neural Networks 10 (1996) 1659–1671.
[35] W. Maass, Fast sigmoidal networks via spiking neurons, Neural Computation 9 (1997) 279–304.
[36] H. Markram, The Blue Brain project, Natural Review of Neuroscience 7 (2006) 153–160.
[37] W. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The Bulletin of Mathematical Biology 5 (1943) 115–133.
[38] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, D.S. Modha, A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45 nm, in: R. Patel, T. Andre, A. Khan (Eds.), Custom Integrated Circuits Conference, CICC, IEEE, 2011, pp. 1–4.
[39] H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, Top500 supercomputing sites, http://www.top500.org/, 2011, [Online; accessed 12-Dec-2011].
[40] A. Mortara, E.A. Vittoz, A communication architecture tailored for analog VLSI artificial neural networks: intrinsic performance and limitations, IEEE Transactions on Neural Networks 5 (1994) 459–466.
[41] J. Navaridas, M. Luján, J. Miguel-Alonso, L.A. Plana, S. Furber, Understanding the interconnection network of SpiNNaker, in: Proceedings of the 23rd International Conference on Supercomputing, ICS'09, ACM, New York, NY, USA, 2009, pp. 286–295.
[42] J. Navaridas, L.A. Plana, J. Miguel-Alonso, M. Luján, S.B. Furber, SpiNNaker: impact of traffic locality, causality and burstiness on the performance of the interconnection network, in: Proceedings of the 7th ACM International Conference on Computing Frontiers, CF'10, ACM, New York, NY, USA, 2010, pp. 11–20.
[43] M. Pearson, I. Gilhespy, K. Gurney, C. Melhuish, B. Mitchinson, M. Nibouche, A. Pipe, A real-time, FPGA based, biologically plausible neural network processor, in: Artificial Neural Networks: Formal Models and Their Applications, ICANN 2005, in: Lecture Notes in Computer Science, vol. 3697, Springer, Berlin/Heidelberg, 2005, pp. 755–756.
[44] P. Pfaerber, K. Asanovic, Parallel neural network training on MultiSpert, in: Proceedings of IEEE 3rd Int'l Conference on Algorithms and Architectures for Parallel Processing, ICA3PP, pp. 659–666.
[45] L. Plana, J. Bainbridge, S. Furber, S. Salisbury, Y. Shi, J. Wu, An on-chip and inter-chip communications network for the SpiNNaker massively-parallel neural net simulator, in: Second ACM/IEEE International Symposium on Networks-on-Chip, 2008, NoCS 2008, pp. 215–216.
[46] L. Plana, S. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, S. Yang, A GALS infrastructure for a massively parallel multiprocessor, Design & Test of Computers, IEEE 24 (2007) 454–463.
[47] A. Rast, F. Galluppi, S. Davies, L. Plana, C. Patterson, T. Sharp, D. Lester, S. Furber, Concurrent heterogeneous neural model simulation on real-time neuromimetic hardware, Neural Networks 24 (2011) 961–978. Multi-Scale, Multi-Modal Neural Modeling and Simulation.
[48] A. Rast, F. Galluppi, X. Jin, S. Furber, The leaky integrate-and-fire neuron: a platform for synaptic model exploration on the SpiNNaker chip, in: The 2010 International Joint Conference on Neural Networks (IJCNN).
[49] A. Rast, X. Jin, M. Khan, S. Furber, The deferred event model for hardware-oriented spiking neural networks, in: M. Kppen, N. Kasabov, G. Coghill (Eds.), Advances in Neuro-Information Processing, in: Lecture Notes in Computer Science, vol. 5507, Springer, Berlin/Heidelberg, 2009, pp. 1057–1064.
[50] T. Sharp, C. Patterson, S. Furber, Distributed configuration of massively-parallel simulation on SpiNNaker neuromorphic hardware, in: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 1099–1105.
[51] Defense Sciences Office, systems of neuromorphic adaptive plastic scalable electronics (SYNAPSE), 2011, [Online; accessed 12-Dec-2011].
[52] P. Worley, Comparison of Cray XT3 and XT4 Scalability, Cray Inc., 2007.
[53] J. Wu, S. Furber, J. Garside, A programmable adaptive router for a GALS parallel system, in: Proceedings of the 2009 15th IEEE Symposium on Asynchronous Circuits and Systems, Async 2009, pp. 23–31.

Cameron Patterson is a Ph.D. student at the University of Manchester whose research interests include real-time visualisation and communications in massively-parallel computing systems. He received a B.Eng. honours degree from the University of Lancaster in the UK in 1996, and prior to undertaking research for his Ph.D. worked in telecommunications. In industry he supported design and maintenance of large scale networks, and he worked in process and product development. Cameron attained Chartered Engineer status in 2003, and has a Cisco CCIE certification in routing and switching (2000).

Jim Garside gained a Ph.D. in Computer Science from Manchester University in 1987 for work in signal processing architecture. Post-doctoral work on parallel processing systems based on Inmos Transputers was followed by a spell in industry writing air traffic control software. Returning to academia gave an opportunity for integrated circuit design work, dominated by design and construction work on asynchronous microprocessors in the 1990s. More recently he has been involved with dynamic hardware compilation, GALS interconnection and the development of the hardware and software of the SpiNNaker neural network simulator.

Eustace Painkras received B.Tech. degree from Kerala University and M.S. degree from State University of New York at Stony Brook. She is a research fellow in the School of Computer Science at the University of Manchester. Her research interests include VLSI design, wireless communications systems, reliable and power-aware computer architectures.

Steve Temple is a Research Fellow in the School of Computer Science at the University of Manchester. His research interests include microprocessor system design and self-timed logic. He has a BA in Computer Science from the University of Cambridge, UK, and a Ph.D. for research into local area networks from the University of Cambridge Computer Laboratory.

Luis A. Plana received his Ingeniero Electrónico degree from Universidad Simón Bolívar, Venezuela, his M.S. degree in electrical engineering from Stanford University, Stanford, CA, and his Ph.D. degree in computer science from Columbia University, New York. He is a Research Fellow with the School of Computer Science, University of Manchester. Before coming to Manchester, he worked at Universidad Politécnica, Venezuela, for over 20 years, where he was a Professor and Head of the Department of Electronic Engineering. His research interests include the design and implementation of systems-on-chip and on-chip interconnect.

Javier Navaridas is a Royal Society Newton International Fellow in the APT group of the University of Manchester. He obtained his MsEng in Computer Engineering in 2005 and his Ph.D. in Computer Engineering in 2009, both from the University of the Basque Country, Spain. His research interests include interconnection networks for parallel and distributed systems, networks on chip for SoCs and Multiprocessors and characterization of application's behaviour. He has developed and maintained various simulation environments for simulation of different processes and systems.

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016

Thomas Sharp is a Ph.D. student with the SpiNNaker group at The University of Manchester. He received an MSc in Computer Science in 2009 from the University of Manchester, and a BSc in the same subject from the University of Sussex in 2008. His primary interest is simulating cortical areas as systems of millions of spiking neurons in order to investigate the relationship between their structure and function.

Steve Furber is Professor of Computer Engineering in the School of Computer Science at the University of Manchester. He received his bachelor's degree in Mathematics in 1974 and his Ph.D. in Aerodynamics in 1980 from the University of Cambridge, England. From 1980 to 1990 he worked in the hardware development group within the R&D department at Acorn Computer Ltd, and was a principal designer of the BBC Microcomputer and the ARM 32-bit RISC microprocessor. At Manchester he leads the Advanced Processor Technologies group with interests in multicore computing, low-power systems and neural systems engineering.

Please cite this article in press as: C. Patterson, et al., Scalable communications for a million-core neural processing architecture, J. Parallel Distrib. Comput. (2012), doi:10.1016/j.jpdc.2012.01.016